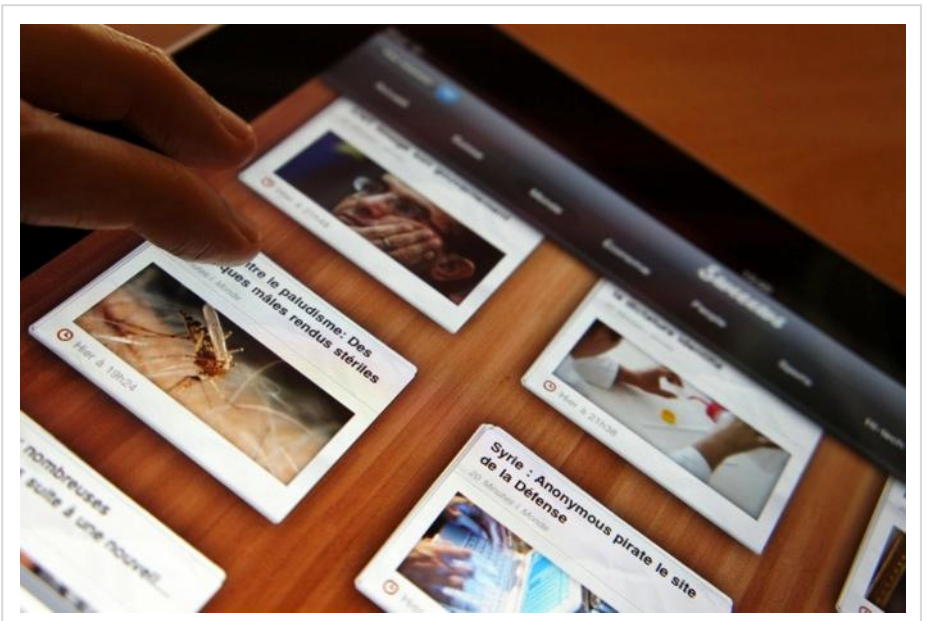
Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de Bachelor 2011

Filière Informatique de gestion

Sweews

Application iPad pour la presse nationale



Etudiant : Jonathan Moreira

Professeur : Florian Doche

RÉSUMÉ

L'actualité est aujourd'hui au centre de notre société numérique. En effet, les tablettes et autres appareils mobiles ont accentué ce phénomène de consommation des médias. Que ce soit à travers un simple journal, un site Internet ou une application native, l'utilisateur a aujourd'hui besoin d'avoir un accès facile et continu à l'information journalistique.

Cette demande est telle que de grandes firmes ont vite compris l'enjeu. Apple va en effet dès l'automne 2011 intégrer une plateforme pour ses appareils mobiles donnant la possibilité à l'utilisateur d'acheter ses journaux préférés à tout moment. Cette solution, nommée Newstand, répond aussi à la demande croissante des éditeurs de presse qui sont aujourd'hui confrontés à une délicate problématique : offrir de l'information tout en dégageant des bénéfices.

Ce travail de Bachelor s'inscrit dans cette problématique. En effet, le but est de développer une application iPad pour la presse nationale suisse. Cette application, appelée Sweews, n'a pas pour ambition de concurrencer le Newstand d'Apple. Au contraire, les objectifs de notre application sont :

- ✓ De proposer un ensemble d'actualités à l'utilisateur suisse romand tout en restant gratuite;
- ✓ De promouvoir l'HES-SO Valais en exposant l'actualité de l'école et plus précisément les projets RAD;
- ✓ De promouvoir les acteurs de la presse nationale en leur permettant de toucher un cercle plus large de lecteurs potentiels.

Pour mener à bien le développement de Sweews, ce projet est géré via la méthodologie SCRUM et s'articule autour de plusieurs étapes. D'une application générique prête à être distribuée à un journal, jusqu'à un « lecteur de flux » capable de récupérer l'actualité des journaux suisses : les directions possibles d'un tel projet sont nombreuses. Il s'agit donc, en premier temps, de définir la direction la plus adaptée aux objectifs de ce projet. Suite à cela, une analyse des applications concurrentes suisses et internationales est indispensable afin de définir les spécificités de l'application et la rendre compétitive.

Cette première phase d'initialisation et d'analyse passée, il s'agit, ensuite, de mettre en place l'architecture MVC de l'application et de commencer le développement de l'application.

TABLE DES MATIÈRES

1 INTRODUCTION

Je commencerai ce rapport par l'introduction de ce travail de Bachelor, en expliquant notamment nos divers choix de direction tout au long du projet. D'une application générique prête à être distribuée à un journal, nous avons finalement opté pour le développement d'une application « lecteur de flux » capable de récupérer l'actualité des journaux suisses. Ce changement de direction nous a amenés à définir les principaux objectifs du projet.

J'enchaînerai ensuite sur la gestion de projet via la méthodologie SCRUM et la planification.

Suite à cela, j'expliquerai les résultats de mon analyse des applications concurrentes. Cette recherche m'a en effet permis de ressortir les forces et faiblesses de ces acteurs tout en définissant progressivement les spécificités de Sweews afin d'en faire une application compétitive.

Les spécificités de l'application définies, j'expliquerai ensuite l'architecture MVC, le modèle de données et le fonctionnement de Sweews. Je présenterai aussi l'affichage de l'application à travers les différents écrans la composant.

Enfin, je terminerai ce rapport par un bilan général et personnel, en évoquant notamment les points positifs et ceux qui demandent à être améliorés.

1.1 Pourquoi « Sweews » ?

Le nom Sweews vient de deux termes anglophones :

- « **Swiss** » qui signifie « suisse »
- « **News** » qui signifie « actualité »

Les deux termes ensemble forment Sweews. En effet, ce travail a pour objectif le développement d'une application iPad, proposant de l'actualité suisse aux utilisateurs possédant un iPad de 1ère ou 2ième génération.

L'actualité proposée sera celle de médias nationaux tels que LeMatin, le 20 Minutes, 24 Heures et Le Nouvelliste, pour n'en citer que quelques-uns. Dans un premier temps Sweews sera axée sur la suisse romande et sera donc en français.

Sweews servira aussi de moyen de promotion pour la HES-SO Valais, qui l'utilisera comme canal de diffusion pour les news de projets RAD de l'institut de développement.

Avant d'avoir défini précisément ce en quoi consiste Sweews, nous avons pris différentes directions/décisions tout au long du projet. Il serait intéressant d'expliquer ces quelques phases et changements de plan.

1.2 Choix et directions du projet

L'idée de base fut de créer une application iPad générique avec un panneau d'administration accessible via un navigateur. Cette application aurait été par la suite distribuée à un journal suisse. Ce dernier aurait eu ainsi un moyen de diffusion sur iPad de ses news, ainsi que d'un outil permettant de gérer le contenu diffusé (importation via PDF, formulaires à remplir, etc.).

Cette direction semblait judicieuse au premier abord. Cependant, au fil du projet, les faiblesses de ce type d'application ont commencé à apparaître.

En effet, les principaux acteurs suisses de la presse ont aujourd'hui chacun leur propre application iPad, il aurait été vain de leur proposer une autre alternative plutôt générique et peu personnelle. De plus, il aurait fallu les former au panneau d'administration qui va de paire avec l'application iPad. Sans compter qu'il serait compliqué de développer une application iPad et une application web en 11 semaines.

De là, un changement de direction a été entamé vers la troisième semaine du projet. Avec Florian, nous avons décidé que l'application se devait d'être plus générale et de toucher un public plus large. Nous avons donc pensé à créer une sorte d'application « lecteur de flux » permettant d'accéder au contenu des principaux journaux romands. En effet, aujourd'hui certains médias suisses mettent à disposition leur actualité via des flux RSS. Cependant, avant la confirmation de ce changement de direction, j'ai dû effectuer des recherches juridiques afin de valider les aspects légaux d'une telle application.

Etant donné que les flux RSS sont proposés pour un usage personnel et non commercial, et que Sweews serait distribuée gratuitement, nous serions donc dans nos droits. Cette direction de projet validée, nous avons ensuite défini les objectifs principaux du projet.

1.3 Objectifs du projet

Les objectifs de ce travail de Bachelor s'articulent principalement autour de trois types distincts : les objectifs généraux et les objectifs technologiques. Arrivent ensuite les objectifs personnels.

1.3.1 Objectifs généraux

Il s'agit principalement d'objectifs liés à l'application et son utilité finale pour les différents acteurs :

- **Analyser les applications existantes**

  Avant de commencer à définir l'architecture de Sweews, il est important d'analyser les applications concurrentes et de comprendre leur fonctionnement, en ressortant les faiblesses et forces de chacune.

- **Promouvoir l'HES-SO Valais**

  Cet objectif annexe a pour but la diffusion de projets de recherche et actualités de la HES-SO Valais, potentiellement intéressants pour certains utilisateurs de l'application.

- **Promouvoir les acteurs de la presse nationale**

  Sweews doit permettre aux différents médias nationaux de toucher un public cible plus large. Par exemple, les fidèles lecteurs du 20 Minutes auraient aussi un accès facile à d'autres médias qu'ils n'ont pas forcément l'habitude de consulter et vice-versa.

- **Modifier en tout temps les sources d'actualité**

  Les sources de news, et plus précisément les flux RSS proposés, doivent pouvoir être mis à jour par un administrateur de la HES-SO Valais en tout temps, sans devoir faire une mise à jour de l'application native. Ce dernier objectif dépend d'un objectif technologique qui permettrait un accès à distance de l'application iPad à un fichier de configuration des sources.

1.3.2 Objectifs technologiques

La plateforme choisie étant l'iOS pour iPad, les objectifs technologiques sont avant tout centrés sur cet environnement d'Apple. Il s'agit principalement de :

- **Développer une application iPad**

  L'application doit pouvoir supporter la dernière version de l'iOS d'Apple (4.3) et fonctionner sur un iPad de 1ère et de 2ième génération.

- **Ergonomie de l'application**

  Afin d'offrir un confort maximal lors de l'utilisation de Sweews, l'application doit supporter toutes les orientations possibles de l'appareil (le mode paysage et le mode portrait). De plus, l'interface doit offrir un confort dès la première utilisation.

- **Accès et parsage des flux RSS des médias nationaux**

  L'application doit pouvoir récupérer les flux RSS des médias nationaux et les parser afin d'en ressortir le contenu (texte, images, liens, etc.).

- **Accès à un fichier distant pour mise à jour des sources et flux proposés**

  Sweews doit pouvoir accéder à un fichier hébergé sur un serveur de la HES-SO Valais afin d'en récupérer le contenu et ainsi mettre à jour les données locales de l'appareil.

1.3.3 Objectifs personnels

Ce sont avant tout des objectifs que je me suis fixé au début. Ils concernent principalement la formation aux technologies et appareils utilisés durant le projet.

- **Apprendre à travailler sur un environnement Mac OSX**

  N'ayant jamais travaillé sur un Mac, il me faut apprendre à utiliser et comprendre le fonctionnement de cet environnement, très différent de Windows.

- **Apprendre l'Objective-C**

  Il est évidemment nécessaire, avant d'entamer le développement, de comprendre le fonctionnement et la syntaxe de l'Objective-C, très différent des autres langages utilisés pendant mon cursus.

- **Apprendre à développer une application iPad**

  Le développement d'application iPad demande certaines connaissances qu'il est essentiel d'avoir dès le début afin de développer correctement et de ne pas perdre du temps.

- **Acquérir de l'expérience en gestion de projet**

  SCRUM étant la méthodologie choisie, je souhaite parfaire mes connaissances dans ce domaine.

2 GESTION DE PROJET

2.1 Environnement de travail

J'ai effectué ce travail principalement en salle de projet au TechnoArk à Sierre. Cette salle a été mise à disposition par l'HES-SO. J'avais mon poste de travail déjà installé et prêt à l'emploi dès le commencement du projet. Il m'a cependant fallu acquérir un Mac, car le développement iOS nécessite un environnement OSX. J'ai aussi dû configurer une machine virtuelle pour l'hébergement de TinyPM et du fichier source des news.

2.1.1 Intervenants du projet

Ce travail de Bachelor est par définition un travail personnel. Cependant, sans l'aide de certaines personnes, il m'aurait été difficile de le mener à bien. Voici une liste non exhaustive des personnes ayant participé au projet :

- **Florian Doche** – Professeur responsable du projet (**florian.doche@hevs.ch**)
- **Guillaume Fort** – développeur iOS (guillaume.fort@hevs.ch)
- **Joelle Mastelic** – Responsable communication (joelle.mastelic@hevs.ch)
- **Jim Dovey** – créateur de l'API AQGridView (jimdovey@mac.com)

Je tiens à souligner la disponibilité de Jim Dovey, vivant au Canada, qui malgré son travail a pris le temps de répondre à mes quelques questions sur son API.

2.1.2 Outils et machines utilisés

J'ai utilisé différents systèmes et outils durant ce projet :

Applications:

- **Xcode 4.0** : IDE pour Mac OSX pour le développement de l'application
- **MesaSQLite** : outil de gestion de BDD SQLite
- **Photoshop** : pour créer le design de l'application
- **Notepad++** : pour créer et modifier le fichier « sources.xml »
- **Simulateur iOS** : pour simuler l'application sur un iPad

Web-applications :

- **TinyPM** : pour la gestion SCRUM du projet
- **Dropbox** : pour l'échange de fichiers entre les différentes machines utilisées

Machines et environnements

- **MacBook Pro** 15' avec Mac OSX Lion pour le développement
- **iPad** 1ère génération pour tester l'application
- **Machine virtuelle avec Windows Server 2003 et IIS** pour l'hébergement du fichier distant et de TinyPM. Cette machine est hébergée en zone démilitarisée sur un serveur du service informatique de la HES-SO.

2.2 Méthodologie de développement

Ayant déjà fait l'expérience de la méthode « SCRUM » lors du projet de groupe de 10 semaines effectué au début de ce semestre, j'ai choisi d'opter pour cette méthodologie que je trouve efficace et réactive.

2.2.1 La méthode SCRUM

Figure 1 - Méthodologie SCRUM

SCRUM est une méthode de gestion de projet faisant partie de la famille des méthodes agiles. Ce type de méthode implique au maximum le client dans le déroulement du projet ce qui permet de satisfaire directement ses besoins plutôt qu'un cahier des charges signé en début de projet.

De ce fait, cette méthode offre une certaine réactivité tout au long d'un projet de développement. Elle permet en outre de modifier en cours de route les spécifications de l'application. Par exemple, une fonctionnalité qui semblait essentielle en début de projet, peut en cours de route devenir optionnelle à la demande du client. C'est ce qui est arrivé avec l'abandon du panneau d'administration.

Cette méthode m'a donc permis de découper mon projet en plusieurs sprints et de réagir à des changements de direction arrivés en cours de développement. J'exposerai plus loin ma planification ainsi que le détail des sprints effectués tout au long du projet.

2.2.2 Outil SCRUM

Pour appliquer cette méthodologie de développement, j'ai utilisé une application web appelée TinyPM.

tinypm

Figure 2 - Outil SCRUM

Cette application, gratuite pour 5 utilisateurs, offre un grand nombre de fonctionnalités :

- Gestion des « user stories »
- Gestion du « product backlog »
- Gestion des tâches
- Gestion de sprint
- Gestion du planning

Figure 3 - TinyPM, gestion des sprints et « user stories »

Outre, toutes ses fonctionnalités, TinyPM permet aussi d'avoir un aperçu de différentes informations, telles que :

- Graphique « burndown » du projet
- Graphique « burndown » d'un sprint
- Décomptes des heures (total et par itération)
- Avancement du projet avec barre de progression

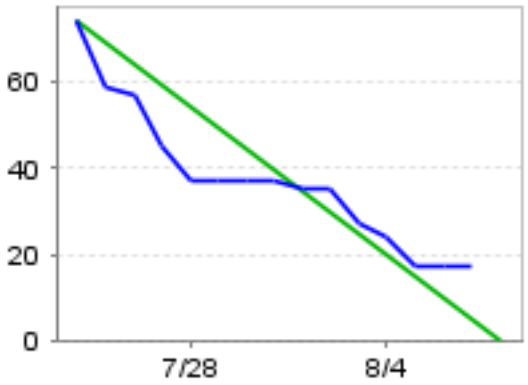


Figure 4 - TinyPM, graphique burndown

Itération	Heures
Sprint 6	108.0
Sprint 5	110.5
Sprint 4	69.0
Sprint 3	48.0
Sprint 2	41.0

Figure 5 - TinyPM, décompte des heures

Ces deux imaCes deux figures ne sont là qu'à titre d'exemples afin de montrer les nombreuses fonctionnalités offertes par cet outil de gestion de projet. Je reviendrai en détail sur la planification et les heures effectuées dans le chapitre suivant.

2.2.3 Réunions

Avec mon responsable de projet, nous avions fixé dès le début une fréquence de 1 réunion tous les 15 jours. Ceci m'a permis en effet de lui donner un feedback détaillé de l'avancement du projet. Il pouvait ainsi me recadrer quand il le fallait.

Ces réunions étaient importantes car elles définissaient la direction du projet. C'est en effet lors de ces discussions que nous avons pris des décisions, notamment lorsque nous avons décidé d'oublier l'idée du panneau d'administration.

Ces réunions duraient approximativement une heure. À la fin de chacune, je rédigeais un procès verbal retraçant les points importants traités. Ensuite, je mettais à jour le product backlog et le sprint en cours, en fonction des décisions prises.

2.3 Planification

Ce travail de Bachelor se déroulant sur 11 semaines du 16 mai 2011 au 16 août 2011, la charge de travail prévue dans le règlement est de 360 heures minimum.

2.3.1 Planification initiale

Afin d'entamer au mieux ce projet, j'ai effectué une planification initiale lors de la première semaine, juste après la définition des spécificités générales du projet.

Pour cette 1ère planification, j'ai opté pour un découpage basé sur la méthode RAD afin d'avoir une vue d'ensemble de la charge de travail qui m'attendait. Elle a permis aussi à mon responsable de projet de recadrer des estimations erronées en fonction des tâches prévues. En effet, j'avais prévu 2 jours pour la rédaction du rapport. Florian m'a fait prendre conscience qu'une telle tâche demande beaucoup plus de temps.

Cette planification a été peu évidente à faire étant donné que le projet et les technologies utilisées m'étaient encore inconnues. De ce fait, il a fallu tenir compte d'un temps d'apprentissage qu'il était difficile d'estimer avant d'avoir réellement commencé. Cette problématique va comme nous allons le voir créer un écart dans certaines étapes par rapport à la planification finale. Voici donc ma planification initiale créée avec MS Project :

Nom de la tâche	Durée	Début	Fin
Sprint 1 (lancement projet)	**11 jours**	**Mar 17.05.11**	**Ven 27.05.11**
Recherche et tests d'applications concurrentes	4 jours	Mar 17.05.11	Ven 20.05.11
Préparation poste de travail	2 jours	Sam 21.05.11	Dim 22.05.11
Installation et configuration logiciels	1 jour	Lun 23.05.11	Lun 23.05.11
Planification initiale	1 jour	Mar 24.05.11	Mar 24.05.11
Rédaction "cahier des charges"	3 jours	Mer 25.05.11	Ven 27.05.11
Sprint 2 (Apprentissage & Design)	**10 jours**	**Lun 30.05.11**	**Ven 10.06.11**
Suivre tutoriel sur création d'application iPad	4 jours	Lun 30.05.11	Jeu 02.06.11
Création et validation prototypes écran	1 jour	Ven 03.06.11	Ven 03.06.11
Design et implémentation interface	3 jours	Lun 06.06.11	Mer 08.06.11
Design BDD	2 jours	Jeu 09.06.11	Ven 10.06.11
Sprint 3 + Sprint 4 (application iPad)	**20 jours**	**Lun 13.06.11**	**Ven 08.07.11**
Développement application iPad	20 jours	Lun 13.06.11	Ven 08.07.11
Sprint 5 + sprint 6 (administration)	**20 jours**	**Lun 11.07.11**	**Ven 05.08.11**
Développement panneau d'administration	20 jours	Lun 11.07.11	Ven 05.08.11
Sprint 7 (Finalisation)	**6 jours**	**Lun 08.08.11**	**Sam 13.08.11**
Tests et corrections des bugs	2 jours	Lun 08.08.11	Mar 09.08.11
Mise en production de l'application	1 jour	Mer 10.08.11	Mer 10.08.11
Rédaction documentation	1 jour	Jeu 11.08.11	Jeu 11.08.11
Rédaction rapport de travail	2 jours	Ven 12.08.11	Sam 13.08.11

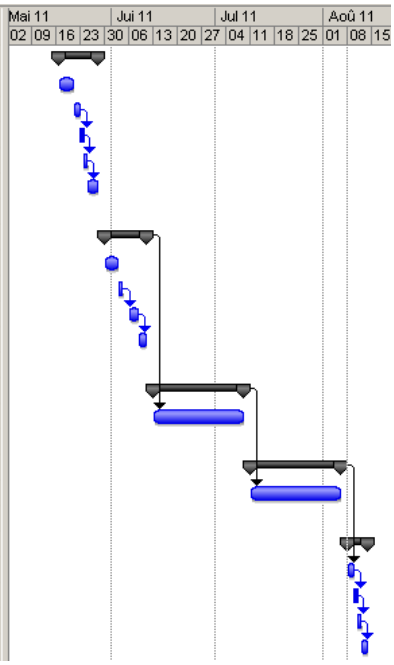


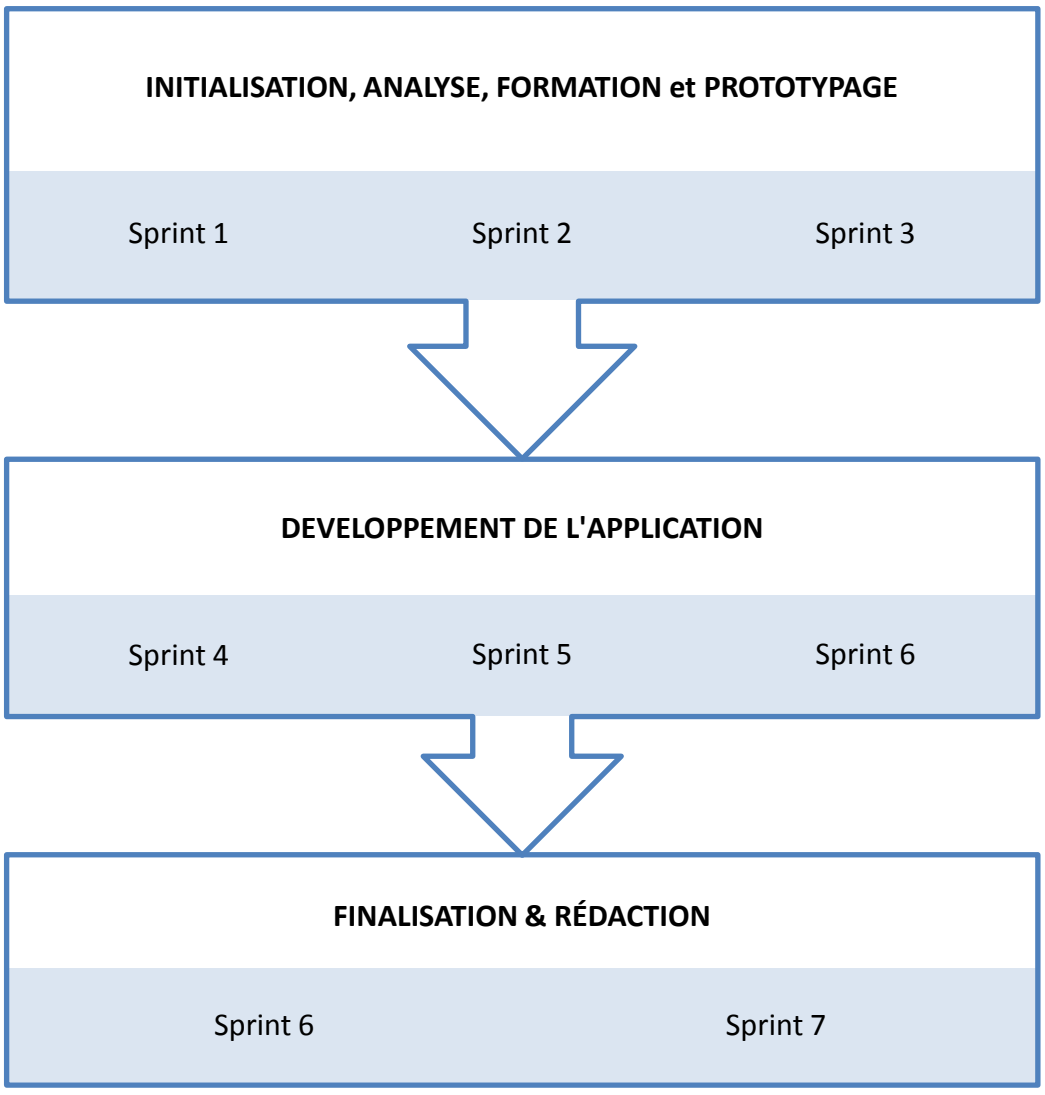
Figure 6 - Planification initiale

2.3.2 Planification finale

La planification effective diffère passablement. Elle se décompose aussi de 7 sprints au total :

- 6 sprints d'une durée de 15 jours
- 1 sprint d'une durée de 7 jours.

Les activités du projet se sont donc décomposées de la manière suivante:



Il s'agit là d'une vue très synthétique du déroulement du projet. Pour plus de détails, voici un tableau récapitulatif des 7 sprints composant le projet :

SPRINT	Date et heures	Résumé
1	16.05.11 –30.05.11 **33 heures**	✓ Installation environnement de travail et serveur avec TinyPM ✓ Recherche et analyse d'applications concurrentes ✓ Configuration outils de travails ✓ Rédaction cahier des charges
2	30.05.11 –13.06.11 **34 heures**	✓ Formation Objective-C ✓ Création architecture ✓ Design prototype écran et design de l'in- ✓ terface
3	13.06.11 –27.06.11 **38 heures**	✓ Formation développement iPad ✓ Installation et prise en main de Xcode 4.0 ✓ Création prototype pour tester faisabilité
4	27.06.11 –10.07.11 **60 heures**	✓ Développement application ✓ Création vue principale ✓ Intégration parseur des flux RSS
5	10.06.11 –25.07.11 **93 heures**	✓ Persistance des données via BDD locale CoreData et ajout de contenu ✓ Navigation des articles par catégorie ✓ Implémentation filtres des news
6	25.07.11 –08.08.11 **108 heures**	✓ Accès au fichier de données à distance ✓ Implémentation rubriques à thème ✓ Implémentation design ✓ Implémentation du système de mis à jour à distance des sources ✓ Tests et corrections de divers bugs
7	08.08.11 –14.08.11 **49 heures**	✓ Rédaction rapport final ✓ Création des délivrables (CD et impression des documents)

Total d'heures effectuées: **415 heures**

Ce récapitulatif se base sur mon suivi des heures via l'outil TinyPM. Pour plus d'informations et de détails sur le déroulement du projet, il est possible de consulter le journal des tâches joint en annexe. Vous pouvez aussi accéder directement à TinyPM. Les coordonnées d'accès se trouvent dans l'annexe « Informations d'accès ».

2.3.3 Bilan de la planification

En comparant les deux planifications, nous remarquons tout de suite que la phase de formation a été évaluée très en dessous de la charge de travail réelle. J'ai en effet passé plus de temps prévu sur la formation et l'apprentissage des technologies utilisées pour ce projet. Dans la planification initiale, j'avais prévu la formation uniquement pour le sprint 2 qui dure 15 jours, or au final j'ai passé les 2 premiers sprints à me former sur les diverses technologies soit sur 30h environ. Sans compter qu'il m'a aussi fallu apprendre à utiliser un environnement Mac, qui m'était jusque-là inconnu.

Ensuite, les changements de direction au sein du projet ont fait que certaines étapes de développement prévues ont été supprimées. Il s'agit du développement du panneau d'administration. Nous voyons là l'utilité de la méthode SCRUM qui m'a permis de réadapter mon planning et mes tâches de manière efficace. De plus, avec recul, je pense qu'il aurait été difficile de développer un panneau d'administration avec le temps imparti. Je pense que cette erreur de planification initiale découle de ce manque d'expérience dans les technologies utilisées, ce qui m'a valu de sous-estimer la phase de développement iOS.

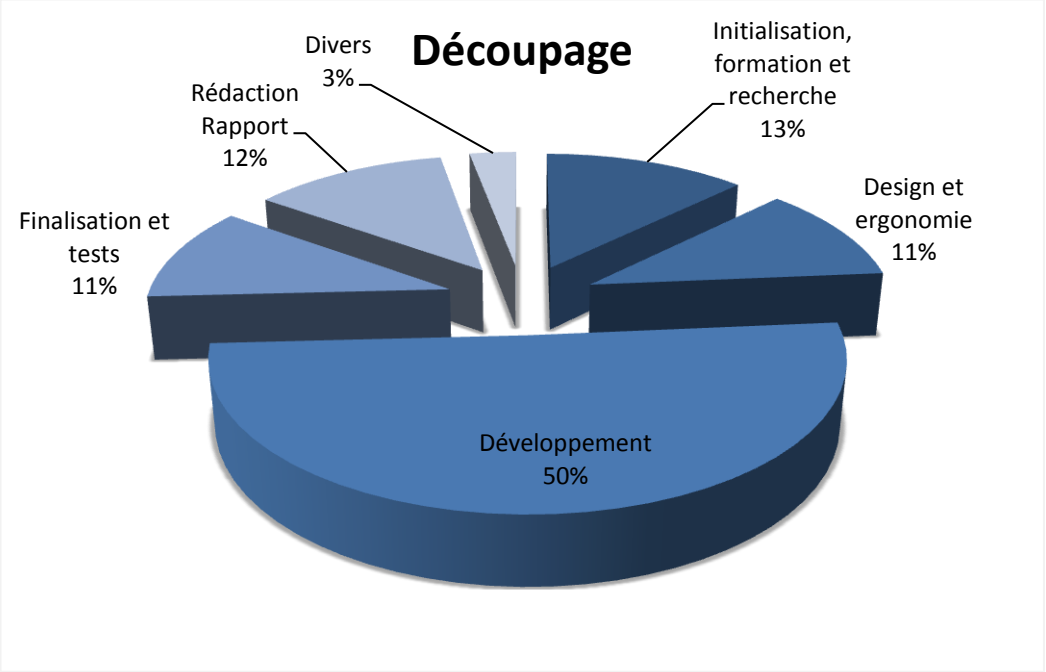


Figure 7 - Découpage des étapes

J'ai en effet passé au total environ 200 heures sur l'implémentation de l'application iPad sur une durée de 4 sprints. Alors que dans ma planification initiale, j'avais prévu 128 heures sur 2 sprints. Je pense que ce total d'heures découle de mon manque d'expérience en début de projet. Il m'arrivait parfois de refaire toute l'implémentation d'une fonctionnalité car je l'avais développée de manière trop complexe ou erronée.

D'ailleurs, si l'on regarde attentivement les graphiques burndown, on remarque que sur les 2 principaux sprints de développement, je suis plus productif sur le cinquième

que sur le quatrième. Cela démontre que j'ai acquis de la facilité au fur et à mesure. J'ai en effet développé plus efficacement dans les 2 derniers sprints que dans les premiers.

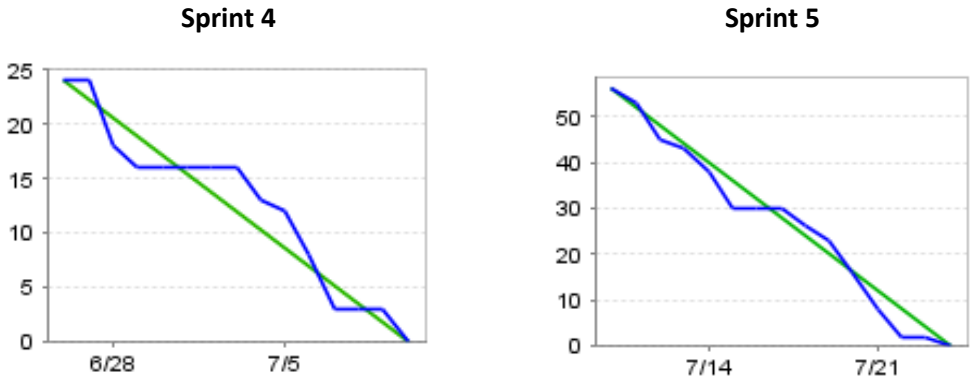


Figure 8 - Graphiques burndown, sprint 4 et sprint 5

Ensuite, certaines tâches ont tout simplement été oubliées lors de la planification initiale. Il s'agit avant tout de tâches dites administratives (réunions, rédaction de procès verbaux) qui m'ont pris au total 15 heures.

Pour conclure ce bilan de planification, je pense que j'ai commis l'erreur de sous-estimer le développement de l'application iPad. Ayant développé bon nombre d'applications web (notamment sur le framework .NET), je pensais que la productivité serait semblable. Il s'est avéré que non. J'ai pris beaucoup plus de temps que prévu pour développer les fonctionnalités et implémenter les vues. Comme je l'ai expliqué cette sous-évaluation découle directement de mon manque de connaissances en Objective-C au début du projet. Pour la petite anecdote illustrative de cette problématique, j'ai passé plus de deux heures à essayer d'afficher une chaîne de caractères dans la console.

Cependant, le point positif à ressortir concerne mon adaptation du plan en cours de route lorsque l'idée du panneau d'administration a été abandonnée. Je pense que la méthode SCRUM m'a permis de réagir efficacement à ce changement soudain. Cela m'a démontré une fois de plus l'efficacité des méthodes agiles dans la gestion de projets informatiques.

3 ANALYSE DE LA CONCURRENCE

3.1 Applications suisses

En Suisse, il existe de nombreux concurrents dans le domaine de l'actualité. Généralement ce sont des journaux qui ont leur propre application :

- 20 Minutes
- LeMatin HD
- SF - schweizer fernsehen
- TSR – Télévision Suisse Romande

Il existe aussi une application indépendante qui récupère les flux d'actualités de différents médias, nationaux et internationaux

- Newsmix de Sobees

3.1.1 LeMatin

L'application LeMatin est intéressante à analyser, notamment son interface, agréable à l'œil certes, mais à mon avis, peu ergonomique :

Figure 9 - L'application LeMatin HD

En effet, il y a trop d'informations différentes sur l'écran principal. Il devient dès lors difficile de s'y retrouver. Il sera important pour Sweews d'éviter ce genre d'interface et de limiter l'information affichée pour plus de clarté dans la navigation.

3.1.2 Newsmix

Newsmix a été développée par Sobees, une entreprise suisse. C'est une application « lecteur de flux » qui récupère l'actualité de différents médias nationaux ou internationaux, ainsi que de réseaux sociaux tels que Facebook et Twitter. Dès le premier lancement de l'application, on sent que Newsmix a été pensée et développée de manière aboutie. Le fait que l'application soit payante n'est pas un hasard.

Personnellement, je trouve cette application plutôt réussie, notamment son interface, simple mais efficace :

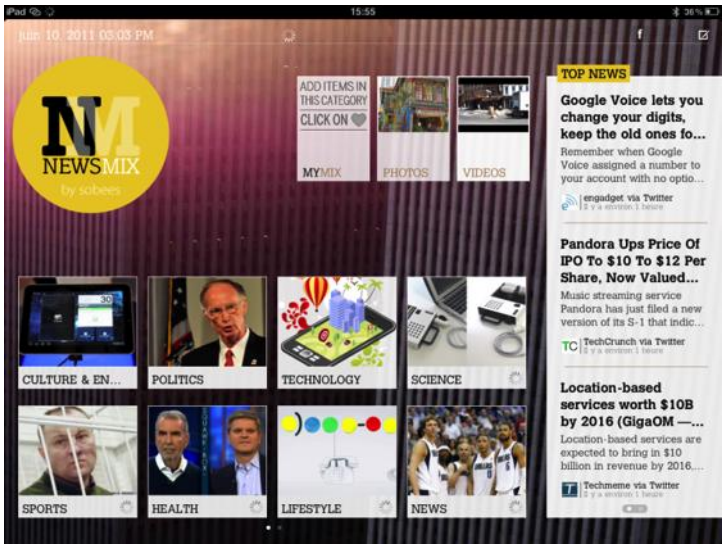


Figure 10 - L'application Newsmix

Autre point positif, l'affichage des vues annexes qui apparaissent en « pop up » au-dessus de la vue principale :

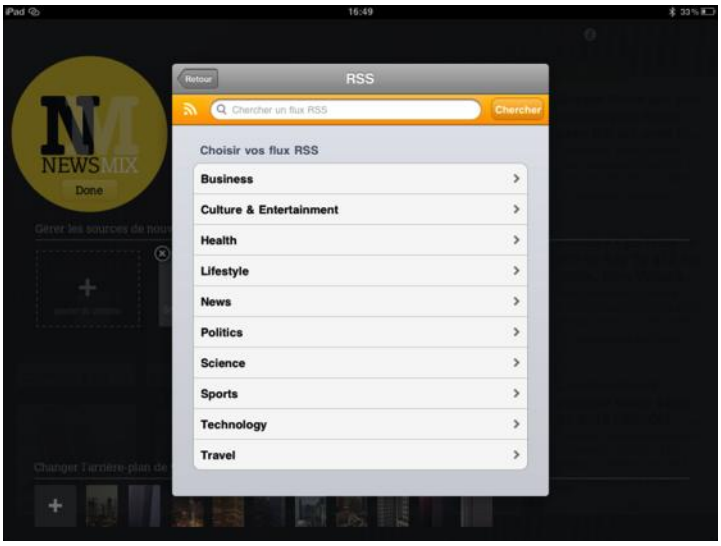


Figure 11 - L'interface de l'application Newsmix

Cette idée semble améliorer la navigation. En effet, dans l'application LeMatin, le changement de vue se faisait en changeant entièrement d'écran, il fallait dès lors revenir en arrière. Ici on garde un aperçu de l'écran principal tout en navigant sur l'écran courant.

Concernant les fonctionnalités, l'idée de lecteur de flux est plutôt intéressante. En effet, grâce à ce système l'utilisateur a accès à un grand nombre de sources différentes. De plus, l'utilisateur peut customiser les sources de news en ajoutant ses propres flux.

Le seul manque de Newsmix concerne le filtrage des sources inexistant. Il faut en effet passer par une phase de configuration pour supprimer/ajouter une source. Il serait judicieux de garder en tête cet élément lors du développement de Sweews.

Newsmix est donc un bon exemple de ce à quoi notre application devrait viser : une application « lecteur de flux » avec interface simple et gestion des sources de news.

3.2 Applications internationales

Au niveau international, il existe plusieurs applications iPad traitant de l'actualité et notamment :

- NewYork Times
- CNN
- BBC
- Financial Times

L'application que je retiendrai parmi celles-ci est l'application de CNN

3.2.1 CNN

Cette application est à mon avis très aboutie, surtout visuellement :

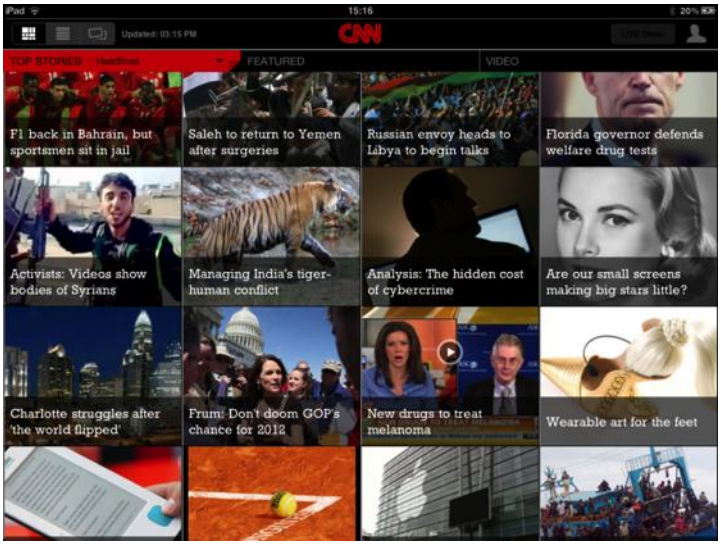


Figure 12 - L'application CNN

Cet affichage par « gridview » des articles est très intéressant, car il permet d'afficher un certain nombres d'articles à la fois. De plus, l'information est plaisante à parcourir. Il serait judicieux de penser à ce type d'affichage pour notre application Sweews.

3.2.2 Flipboard

Flipboard ressemble à l'application suisse Newsmix. Il s'agit en effet d'un lecteur de flux affichant l'actualité de différents médias. Cette application permet aussi de gérer ses propres sources d'actualité :

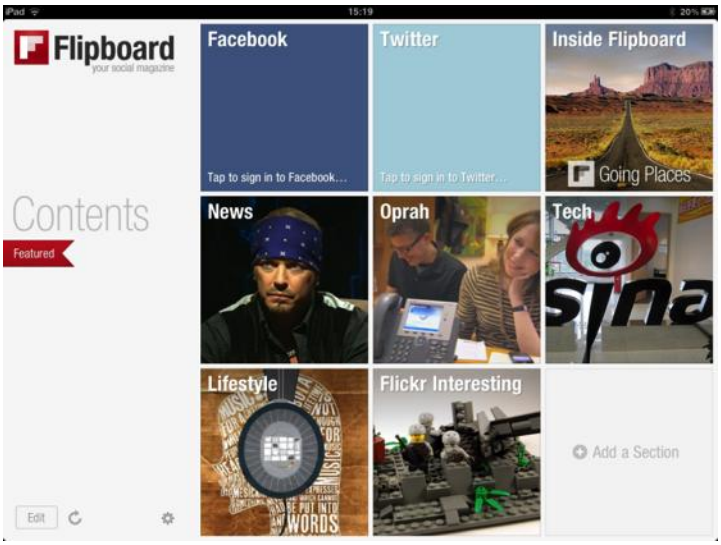


Figure 13 - L'application Flipboard

La force de Flipboard réside dans sa navigation et la possibilité de gérer ses propres sources d'actualité. Comme nous le constatons sur la figure ci-dessus, Flipboard jouit d'une interface sobre et efficace. Le contenu ressort en effet de manière structurée et claire.

Cependant, cette application a elle aussi un point négatif : le manque de filtrage des sources. Il faut en effet ajouter supprimer manuellement des sections. Il aurait été plus simple pour l'utilisateur de pouvoir filtrer en temps réel toutes les news.

3.3 Conclusion de l'analyse

Pour terminer cette analyse, nous remarquons qu'il existe un bon nombre d'applications concurrentes. Cependant, il n'y en a que quelques-unes qui sortent réellement du lot, que ce soit par leurs fonctionnalités ou interface.

En effet, Newsmix et Flipboard sont à mon sens les plus intéressantes. Leur idée de « lecteur de flux d'actualité » est plutôt bien pensée, car elle donne la possibilité de lire différents types d'actualité venant de plusieurs sources.

Les applications appartenant à un média (CNN, LeMatin, 20 Minutes, etc.) limitent quant à elles les sources d'informations étant donné que la seule source est justement celle du média en question. Il faut dès lors télécharger une tierce application d'un autre média si l'on désire avoir accès à d'autres actualités.

En se basant sur ces éléments, il serait intéressant pour notre application qu'elle fonctionne sur le même concept de « lecteur de flux ».

Les raisons de ce choix sont liées aux objectifs principaux du projet:

- Meilleure promotion la HES-SO Valais car groupe de lecteurs plus large
- Meilleure promotion pour les médias nationaux car possibilité d'attirer de nouveaux lecteurs

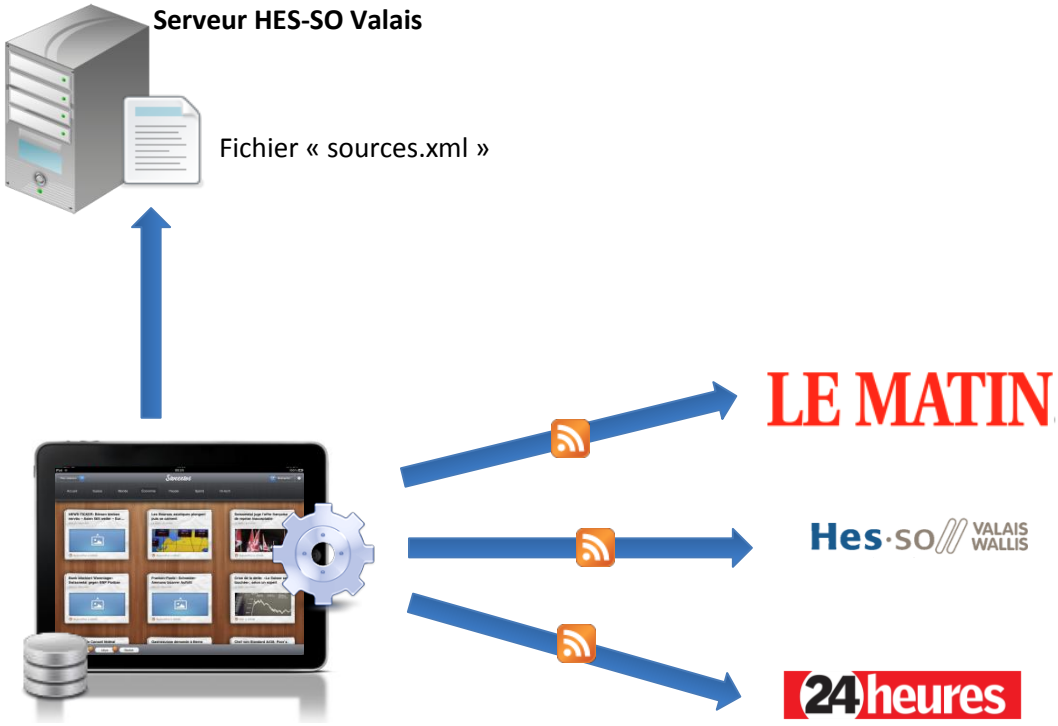
Ensuite, une fonctionnalité importante qui distinguera Sweews de ces deux concurrents est le filtrage en temps réel des sources par catégorie. Cette fonction permettra en effet de choisir et d'afficher rapidement l'actualité d'un média national.

Enfin, quelques-unes de ces applications concurrentes nous permettent de définir les spécificités de la future interface de Sweews. En effet, l'affichage par grille des articles est très intéressante et semble efficace pour ce type d'applications. Il faut aussi retenir l'affichage en « pop up » lui aussi améliorant l'ergonomie générale de l'application.

4 L'APPLICATION

4.1 Architecture

Globalement, l'application Sweews se base sur l'architecture suivante :



Flux RSS des journaux :

- L'application récupère les flux RSS des journaux pour les parser.

Le fichier « sources.xml » est le fichier distant hébergé sur un serveur de la HES-SO Valais. Ce fichier permet de remplir la base de données locale de l'iPad avec les informations suivantes :

- Les sources de news que nous proposons : LeMatin, 20 Minutes, 24 Heures, Le Nouvelliste, et la HES-SO.
- Les différents flux de chaque source pour chaque catégorie
- Les catégories d'actualité
- Les rubriques proposées de base
- Ainsi que les paramètres de l'application telle que la date de mise à jour du fichier source.

Ce fichier servira aussi à mettre à jour les iPads l'ayant déjà utilisé au préalable. La date de mise à jour du fichier est donc primordiale pour indiquer si une mise à jour a été effectuée dernièrement.

4.1.1 MVC

L'architecture interne de l'application iPad est basée sur une architecture MVC :

- Modèles
- Vues
- Contrôleurs

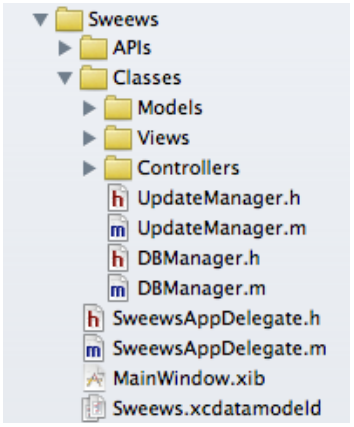


Figure 14 - Architecture MVC

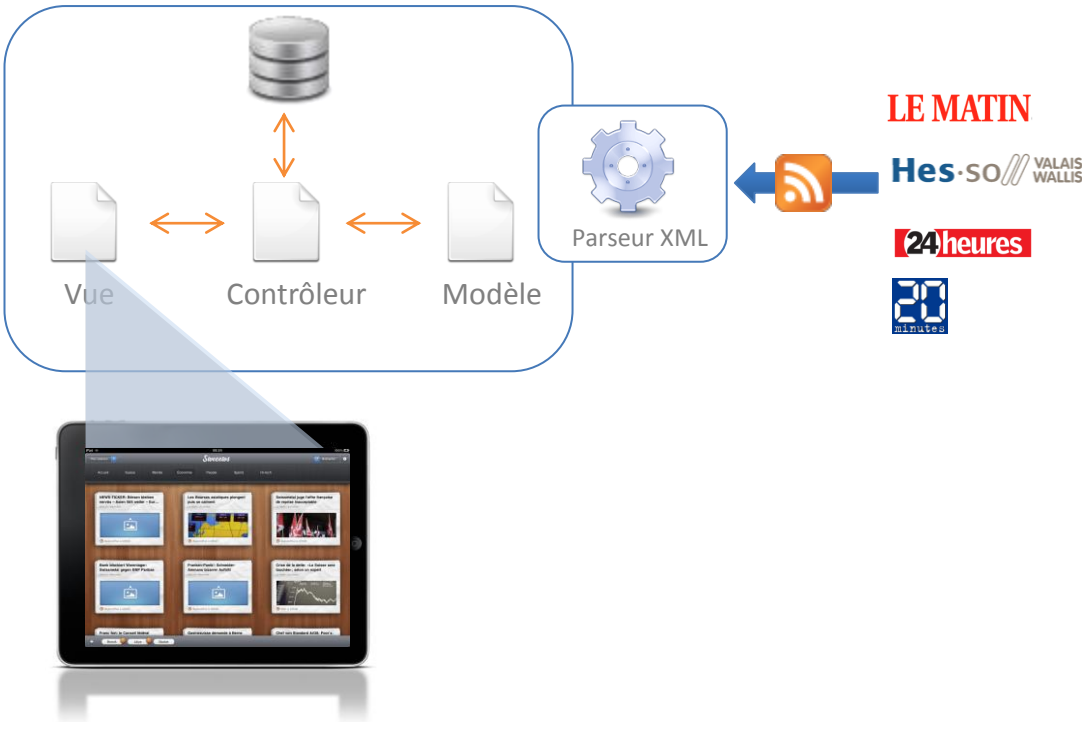
Les modèles sont les entités composant les données manipulées par l'application soit chaque objet utilisé par l'application. Sweews en comporte 6 :

Je reviendrai plus en détail sur chacune de ces entités dans le chapitre expliquant le modèle de données.

Les vues servent, elles, à présenter l'information à l'utilisateur. Ce sont les écrans avec lesquels l'utilisateur va interagir. Sweews en contient un certain nombre, je les présenterai aussi plus en détails dans le chapitre décrivant le fonctionnement de l'application.

Les contrôleurs prennent en charge la gestion des informations affichées via les vues. Ils réagissent en fonction des interactions de l'utilisateur et exécute les fonctions nécessaires pour remplir la demande de l'utilisateur. Ce sont eux qui manipulent les modèles afin de les présenter aux vues.

Voici une illustration de l'architecture interne de l'application :



Pour résumer le fonctionnement de cette architecture, le parseur XML extrait l'actualité des flux RSS récupérés depuis le web. Il va ainsi créer les instances de l'objet du modèle « Article ». Ces objets vont être manipulés par un contrôleur qui va les afficher à travers une vue. Le contrôleur doit parfois récupérer des objets de la base de données locale afin d'afficher les bons articles.

Exemple : pour afficher les articles de la catégorie « Sport », le contrôleur va devoir récupérer au préalable cette catégorie depuis la base de données qu'il va comparer avec tous les articles chargés depuis les flux associés à cette catégorie. Les articles faisant partie de ces flux seront au final affichés dans la vue.

4.1.2 APIs & composants

Sweews a été développé depuis zéro. Cependant, j'ai parfois dû intégrer des composants/API pour implémenter certaines fonctionnalités :

- **AQGridView de Jim Dovey**

  Cette API m'a permis de mettre en place la « gridview » affichant les articles dans la vue principale comme montré dans la figure qui suit. En effet, de base l'API d'Apple pour iPad ne propose pas une classe permettant de créer ce type de vue.

Figure 15 - Gridview affichant les articles

- **NSDateFormater de Michael Waterfall**

  Classe permettant de formater les dates parsées des flux RSS et les convertir en type NSDate.

- **NSArray+Extras.h de Ray Wenderlich**

  Classe permettant de trier par date les articles parsés.

- **ASIHTTPRequest de Ben Copsey**

  API permettant de faire des requêtes HTTP asynchrones. J'ai utilisé cette API pour le chargement et le parsage des flux RSS.

- **GDataXML de Google**

  Cette API de google m'a permis de parser de manière simplifiée les flux RSS et de récupérer leur contenu.

- **GradientButtons de Jeff LaMarche**

  Ce petit composant m'a permis de créer des boutons avec dégradé pour les rubriques.

Malgré ce que l'on peut penser, tout ce travail d'intégration n'a pas été de tout repos. En effet, certains composants étaient plutôt difficiles à implémenter car trop peu documentés. Par exemple, pour l'AQGridView, j'ai dû contacter Jim Dovey par mail afin de réussir à régler un souci de chargement d'images via son API. J'avais essayé de corriger le problème tout seul, mais après une journée de recherche, j'ai dû me résoudre à le joindre. Lui-même a affirmé que son API manquait cruellement de documentation.

Mon autre principal problème d'intégration se résumait aux requêtes asynchrones qu'il a fallu adapter à mon application. J'étais en effet parti d'un tutoriel[1] qui implémentait l'API de Ben Copsey. Ce tutoriel expliquait comment implémenter un lecteur de flux basique pour iPhone. Il n'était donc pas adapté à l'iPad. Il utilisait notamment une « tableview », alors que pour mon application j'avais besoin d'une « gridview ». Les adaptations ont donc été nombreuses.

4.1.3 Base de données et persistance

Afin de persister les données de l'utilisateur entre chaque utilisation de l'application, j'ai dû mettre en place un système. Deux solutions s'offraient à moi :

- Enregistrement dans un fichier p-list (format XML)
- Enregistrement dans une base de données locale (CoreData)

J'ai donc testé les deux alternatives. La première avait pour avantage la gestion directe du fichier via l'éditeur Xcode. Cependant elle était bien trop laborieuse à mettre en place, notamment les algorithmes de sauvegarde qui devaient être adaptés aux modèles, qui eux devaient être créés manuellement. De ce fait, lorsque je modifiais un modèle, je devais dès lors adapter tout mon algorithme pour qu'il fonctionne. Ce moyen me semblait néanmoins suffisant pour les données que je devais persister, jusqu'à ce que je teste la deuxième solution.

La création d'une base de donnée locale pour un iPad comme un iPhone se fait via l'utilisation de l'API CoreData. Cette dernière permet en effet d'organiser les données par des relations « entitées – attributs » en sérialisant les objets. Ces objets peuvent être ensuite stockés en XML, binaire ou SQLite. N'ayant encore jamais travaillé avec des bases de données SQLite, j'ai opté pour cette dernière alternative.

La force de cette solution réside dans la génération automatique des modèles. En effet, si je modifie un attribut d'un objet ou que j'associe une entité comme attribut d'un autre objet, le modèle correspondant peut être généré avec les liens le composant.

[1] Cf. « How To Make A Simple RSS Reader iPhone App Tutorial » de Ray Wenderlich, http://www.raywenderlich.com/2636/how-to-make-a-simple-rss-reader-iphone-app-tutorial

Il s'agit principalement de sauvegarder en mémoire les données de l'utilisateur soit :

- Le choix de ses sources
- Le filtrage des sources
- Les rubriques

Voici un schéma complet de l'architecture de la base de données et des objets manipulés :

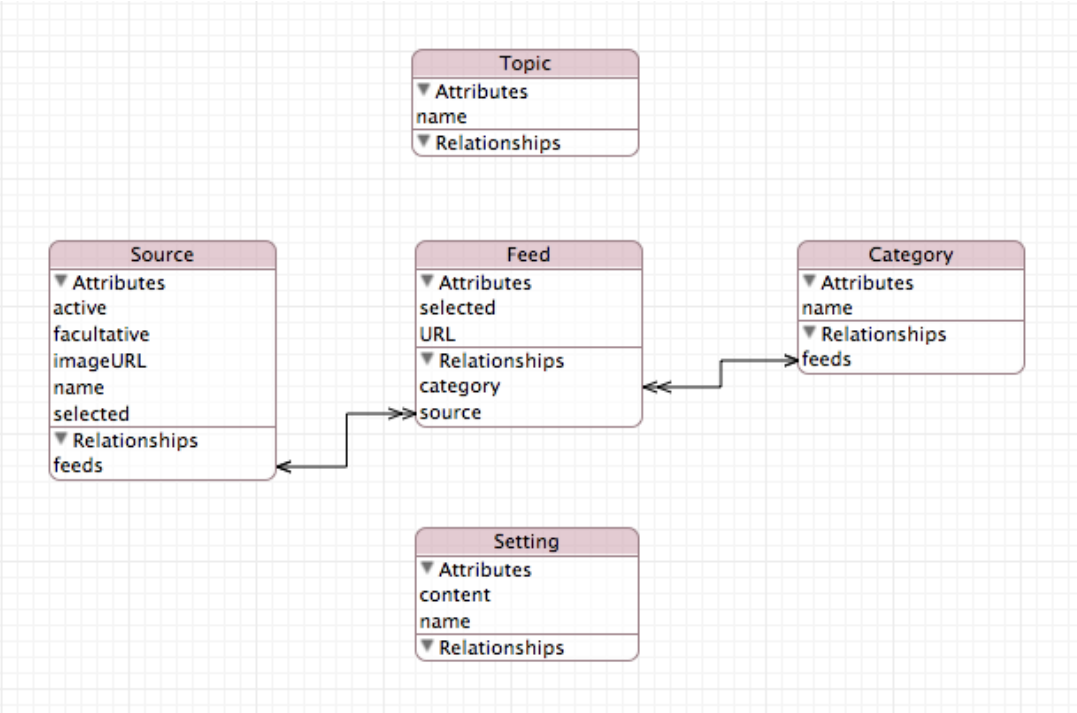


Figure 16 - Architecture de la base de données

Les liens

Les principaux liens entre les entités sont représentés par le pattern « One-to-Many / Many-to-One » :

- Une source contient une liste de flux et un flux est associé à une source.
- Un flux est associé à une catégorie et inversement une catégorie contient une liste de flux.

4.1.4 Modèle de données

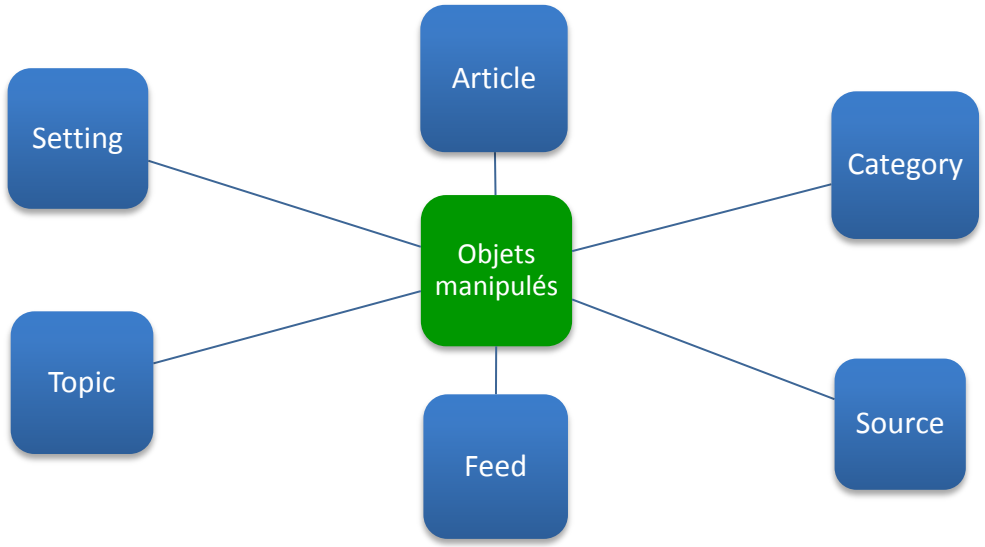


Figure 17 - Les modèles de l'application

Les objets composant le modèle de données ont chacun une utilité bien précise. Voici l'explication de leur architecture respective :

- **Setting**

Paramètres de l'application. Ce peut être la date de mise à jour du fichier source, du texte à afficher dans les messages d'avertissement, etc.

  - **name** : nom du paramètre
  - **content** : contenu du paramètre

- **Topic**

Ce sont les rubriques que l'utilisateur peut créer (ex.: « DSK », « Basketball », etc.) :

  - **name** : nom de la rubrique

- **Category**

Ce sont les catégories de news qui permettent de naviguer entre les différents flux (ex.: « Sports », « Économie », etc.)

  - **name** : nom de la catégorie
  - **feeds** : liste de flux associés à cette catégorie

- **Source**

Ce sont les sources de news, soit les journaux que nous proposons (ex.: LeMatin, 20 Minutes, etc.)

  - **name** : nom du journal
  - **active** : détermine si la source a été activée par l'utilisateur ou non
  - **facultative** : détermine si la source est facultative ou non ; si oui l'utilisateur ne peut la sélectionner/désélectionner et activer/désactiver. C'est le cas par exemple de la source « HES-SO Valais ».
  - **imageURL** : chemin de l'image de la source (ex. : logo du 20 Minutes)
  - **selected** : détermine si la source a été sélectionnée/désélectionnée dans les **filtres**
  - **feeds** : liste de flux associés à cette catégorie

- **Feed**

Ce sont les flux RSS de chaque news pour chaque catégorie

  - **selected** : détermine si la source a été sélectionnée pour la catégorie **respective**
  - **URL** : lien HTTP pointant vers le flux RSS
  - **category** : un flux est associé à une catégorie
  - **source** : un flux est associé à une source de news

- **Article**

Il s'agit du modèle permettant de manipuler les articles chargés depuis les flux RSS. La seule différence avec les 5 autres objets, est qu'il n'est pas persisté. En effet, les articles sont chargés à la volée au lancement de l'application, ou lorsque l'utilisateur rafraîchit les news. De ce fait, à chaque fermeture de l'application les articles sont effacés. Cet objet n'a donc pas de table associée dans la base de données. Il a cependant lui aussi des attributs :

  - **feed** : le flux RSS associé à cet article
  - **newspaperTitle** : le titre de la source (ex. : LeMatin)
  - **articleTitle** : le titre de l'article
  - **articleUrl** : le lien vers l'article
  - **articleImg** : le lien vers l'image de l'article
  - **articleText** : le texte de l'article
  - **articleDate** : la date de l'article

4.2 Fonctionnalités

Il y a deux types de fonctionnalités, celles dites « utilisateur », qui se déclenchent lors de ses actions et les autres plutôt sous-jacentes qui interviennent en arrière-plan.

4.2.1 Utilisateur

Ces fonctionnalités dépendent de l'utilisateur, car elles doivent être déclenchées manuellement :

- Navigation à travers les articles
- Navigation par catégorie
- Affichage d'un article
- Rafraîchissement des news
- Filtrage des sources de news
    - Activation/Désactivation
- Gestion des sources de news
    - Sélection/déselection
    - Filtrage des sources par catégorie
- Gestion des rubriques
    - Ajout/Suppression de rubriques
    - Ajout en série
    - Notifications des rubriques
- Filtrage par rubrique
- Support de l'orientation de l'iPad
- En mode paysage et portrait

4.2.2 Sous-jacentes

Ces fonctionnalités sont propres au fonctionnement de l'application. Elles ne sont pas visibles pour l'utilisateur mais interviennent en arrière-plan.

- Parsage des flux RSS
- Persistance des données
- Système de population de la base de données au 1er lancement
    - Via le fichier XML hébergé à distance
- Système de détection de mise à jour
    - Si détectée, demande de mise à jour
    - Mise à jour des données
- Gestion des avertissements mémoires

4.2.3 Non implémentées

Il s'agit là de fonctionnalités non développées contenues dans la sandbox de TinyPM :

- Recherche et ajout de flux RSS non proposées de base
- Importation de contenu vidéo
- Génération nuage de tags pour création d'une rubrique
- Notifications à l'utilisateur

Ces fonctionnalités n'ont pas été implémentées par manque de temps, mais aussi parce qu'elles n'étaient pas prioritaires par rapport aux autres citées ci-dessus. Cependant, certaines mériteraient d'être conservées pour une amélioration future de l'application.

4.3 Fonctionnement

4.3.1 Cycle de vie de l'application

Sweews a un cycle de fonctionnement à plusieurs étapes. Pour expliquer au mieux le processus de ce cycle, je vais séparer et exposer ces 2 étapes principales :

- Étape 1 : Lancement de l'application
- Étape 2 : Initialisation

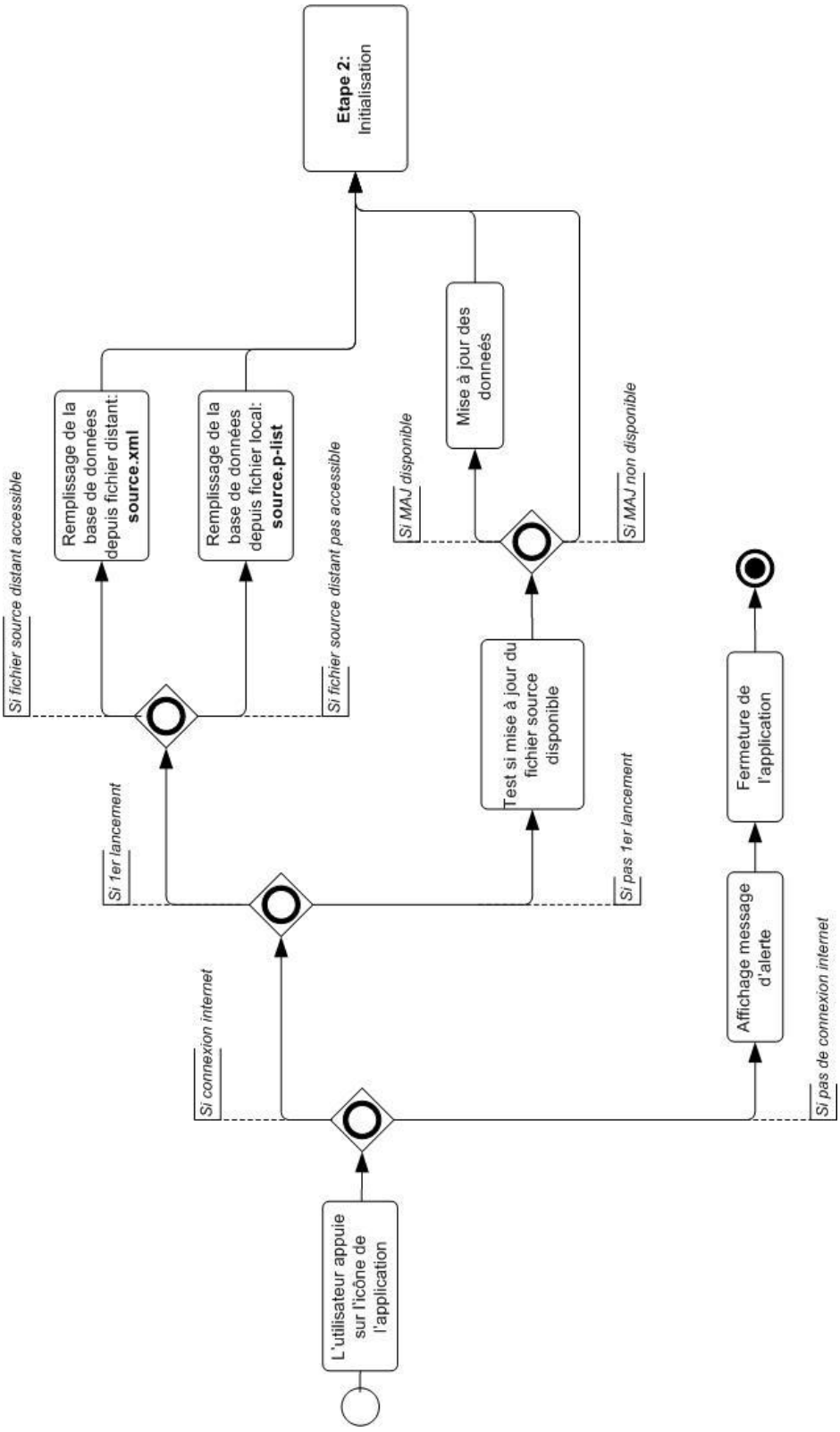
En effet, avant de pouvoir utiliser l'application convenablement, il a fallu mettre en place tout un processus de lancement et d'initialisation des données et éléments graphiques. Ce processus se compose d'un ensemble d'algorithmes qui fonctionnent à la suite, en fonction de certains tests. En effet, comme on va le voir ci-dessous, rien que dans la phase de lancement on compte quatre tests déterminant le lancement de l'application :

1. Tester si l'iPad a une connexion Internet. Sans connexion l'application ne peut fonctionner, donc on lance un message d'alerte et on force la fermeture.
2. Tester si c'est la première fois que l'utilisateur lance l'application.
    a. Si c'est la première fois, il va falloir remplir la base de données. Pour ce faire, on teste si le fichier distant est accessible. S'il ne l'est pas on remplit la base de données avec le fichier local.
3. Si ce n'est pas la première fois que l'application est lancée, on teste qu'il n'y ait pas une mise à jour distante des données.
    a. Et si une mise à jour est disponible, on demande à l'utilisateur s'il désire faire la mise à jour.
    b. S'il n'y a pas de mise à jour, on passe directement à l'étape 2 d'initialisation.

Tout ceci a été laborieux à mettre en place, car il a fallu jongler entre ces différents tests. Afin de bien comprendre en détail comment fonctionne ce processus, j'ai modélisé un schéma pour chacune de ces étapes. Vous les trouverez dans les deux pages suivantes.

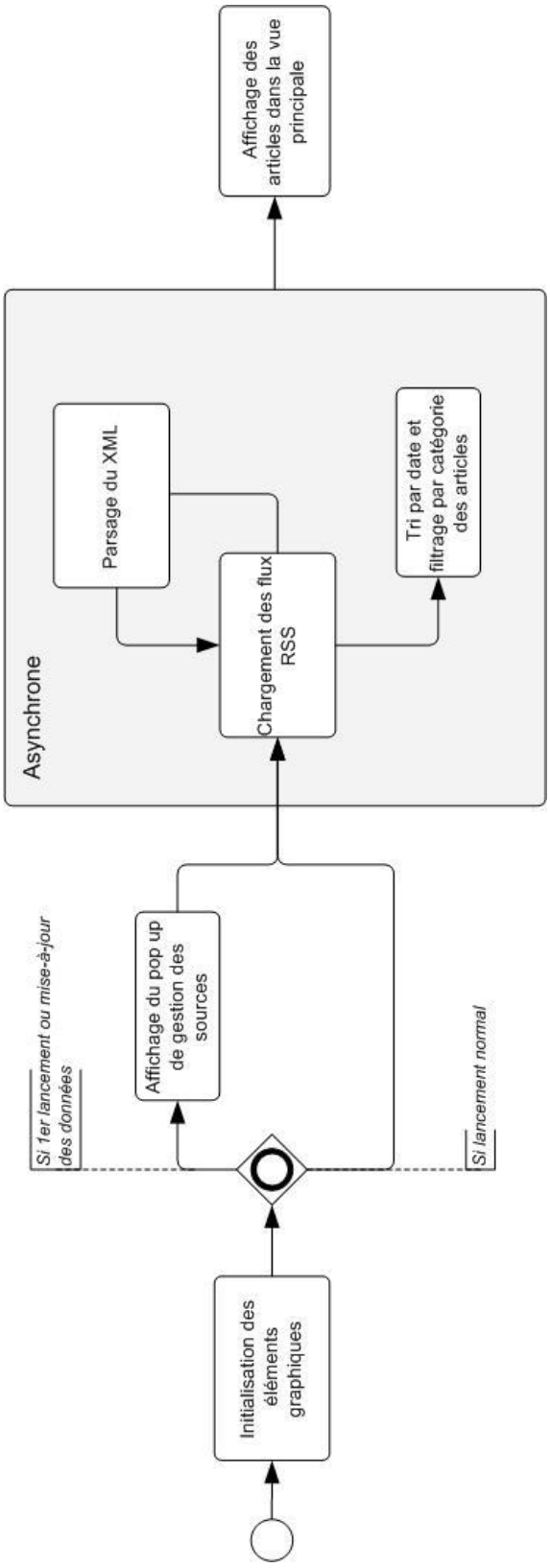
- **Étape 1 : Lancement de l'application**

Ce processus décrit le lancement de l'application, depuis le moment où l'utilisateur appuie sur l'icône de l'application jusqu'à l'initialisation, soit l'étape 2.



- **Étape 2 : Initialisation de l'application**

Cette étape permet d'initialiser les éléments graphiques, de charger/parser les flux RSS, et filtrer les articles afin afficher ceux demandés par l'utilisateur. Une fois cette étape terminée, l'application est prête à être utilisée.



4.3.2 Écrans

Pour pouvoir afficher les articles et autres informations à l'utilisateur, l'application utilise les vues. Ce sont, comme nous l'avons vu dans l'architecture MVC, la partie visible de l'application. Sweews se compose d'une **vue principale**, de vues dites **annexes** ainsi que de vues **d'état**.

- **La vue principale :**

Figure 18 - Vue principale affichant les articles

Il s'agit de la « gridview » affichant les articles des journaux. L'utilisateur aura tout au long du cycle de vie de l'application cette vue devant les yeux. L'utilisateur a plusieurs actions possibles à sa disposition dans cet écran :

- Il peut cliquer sur le bouton « Vos sources » afin de faire apparaître les filtres.
- S'il clique sur le bouton « Modifier », il fait apparaître l'écran de gestion des sources.
- Il peut filtrer les articles en sélectionnant une rubrique située au bas de l'écran.
- En cliquant sur le bouton représenté par un « + », il fait apparaître l'écran de gestion des rubriques.
- Il peut naviguer à travers les catégories via le menu du haut.
- Il peut rafraîchir les news en cliquant sur le bouton « Rafraîchir ».
- Il peut afficher la vue annexe affichant les informations de l'application en cliquant sur le bouton « i ».

- **Les écrans annexes**

Ces écrans interviennent uniquement lors d'un événement ou d'une action de l'utilisateur. Afin d'optimiser l'ergonomie de l'application, j'ai opté pour des vues s'affichant en « pop up » au-dessus de la vue principale, comme l'application Newsmix analysée au début du rapport.

- **L'écran d'affichage de l'article**

Figure 19 - Écran d'affichage de l'article

Cet écran est en fait une « webview » qui charge directement la page internet de l'article sélectionné par l'utilisateur.

- **Écrans de gestion des sources de news et filtres**

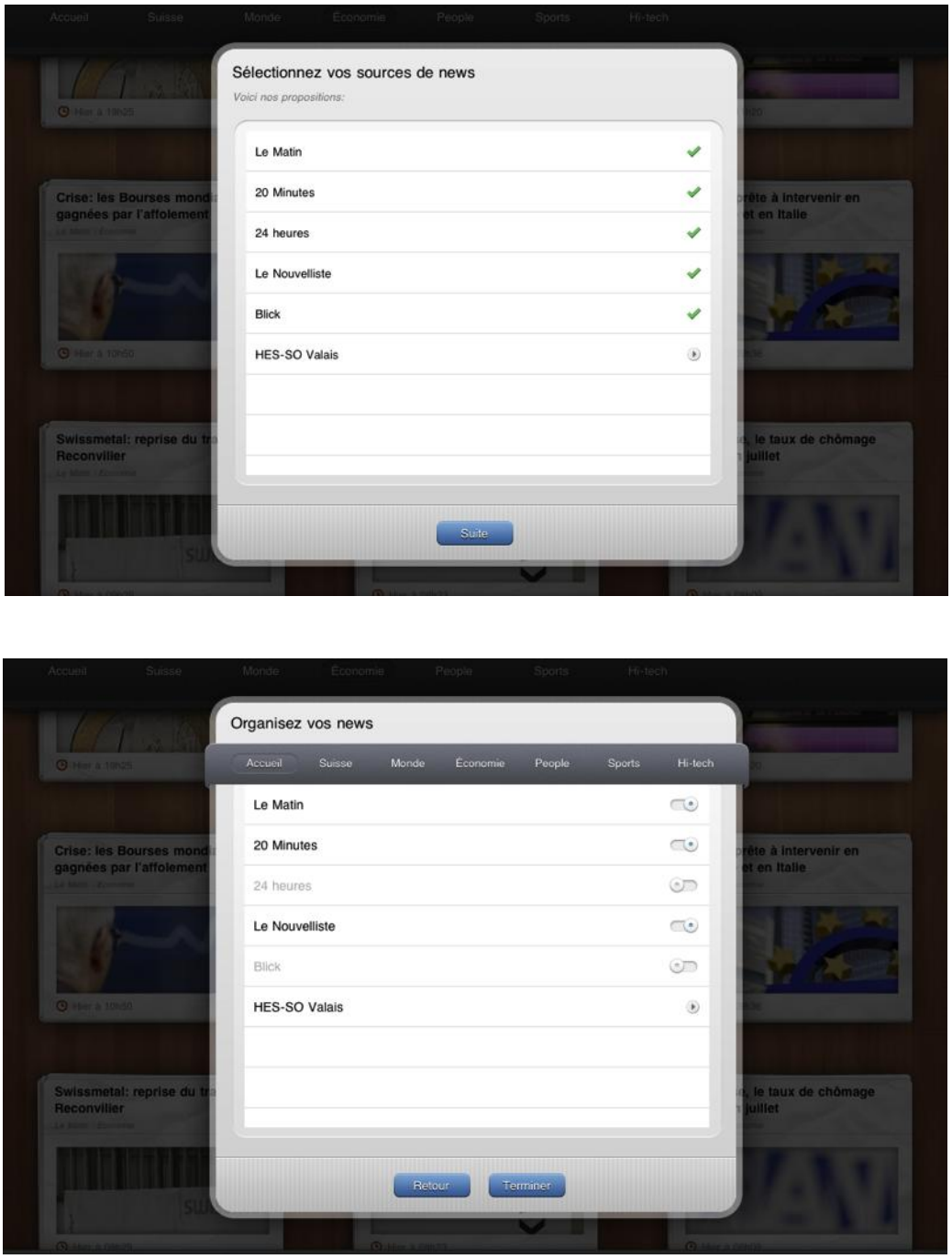
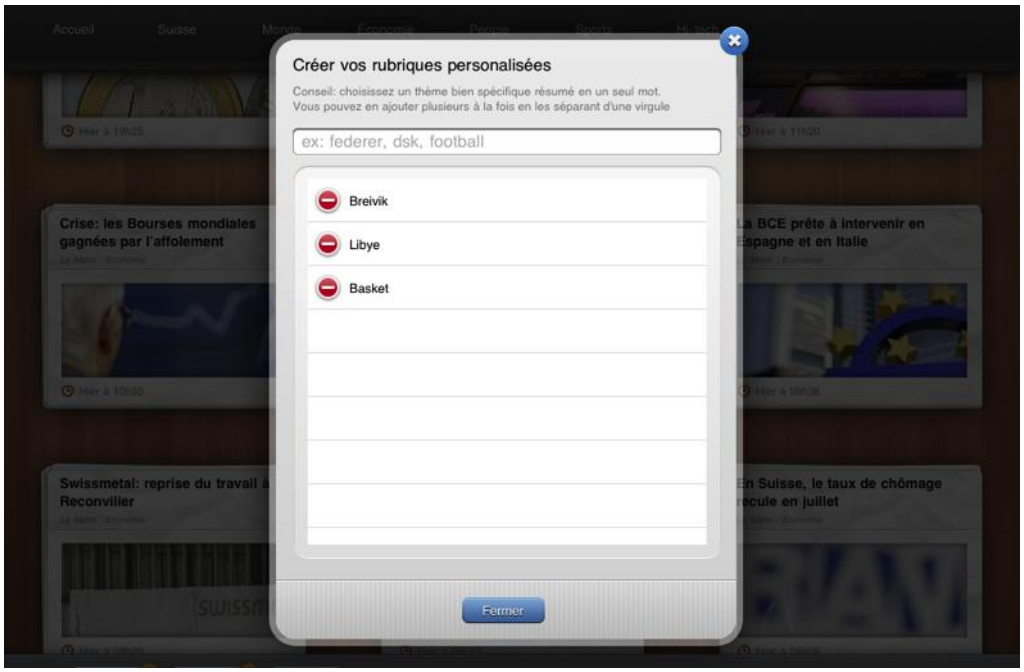


Figure 20 - Écran de gestion des sources et filtres

Dans l'écran du dessus, l'utilisateur peut gérer les sources de news en les activant/désactivant. Dans le deuxième écran, Il peut organiser ses filtres en fonction de chaque catégorie et source activées dans le premier écran. En cliquant sur « Retour », il peut à tout moment revenir sur le premier écran pour faire des modifications.

- **L'écran de gestion des rubriques :**



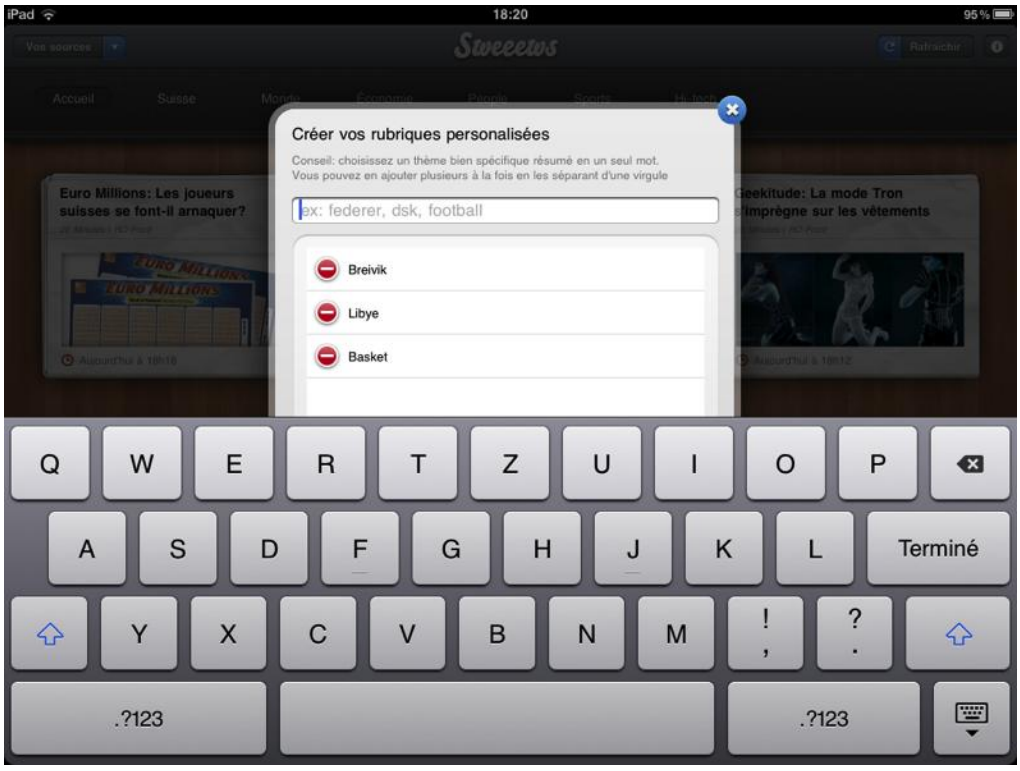


Figure 21 - Écran de gestion des rubriques

Cet écran permet de créer des rubriques par ajout simple ou multiple. En effet, en entrant dans le champ texte plusieurs mots séparés d'une virgule, l'utilisateur va générer plusieurs rubriques en une seule fois. Il peut aussi évidemment supprimer les rubriques en appuyant sur le bouton rouge.

- **L'écran d'affichage des informations de l'application :** cet écran permet d'afficher les informations liées à l'application.

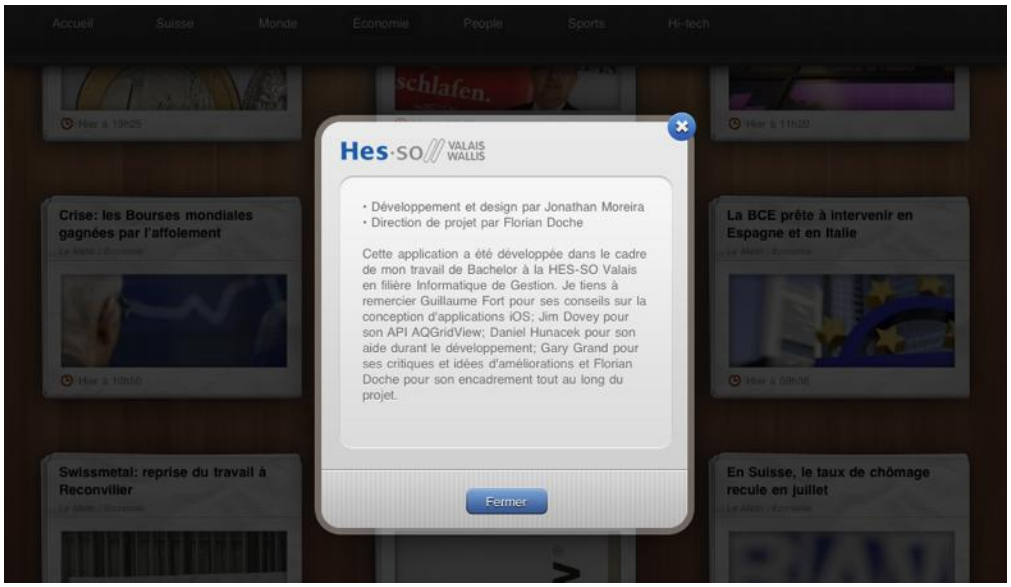


Figure 22 - Écran d'information

- **Les écrans d'état et d'avertissement**

Ces écrans sont là pour indiquer à l'utilisateur l'état de l'application. Ce peut être par exemple un écran indiquant que le chargement des flux RSS est en train de se faire, ou tout simplement un avertissement.

Figure 23 - Message d'alerte

Figure 24 - Message d'avertissement

Tous ces écrans (annexes et d'état) ont été créés de manière à s'adapter à l'orientation de l'iPad. Que ce soit en mode paysage ou portrait, l'affichage se fait au bon format.

NOTE : Pour conclure ce chapitre, je vous invite à regarder la petite vidéo se trouvant sur le CD joint à la fin de ce dossier. J'ai filmé l'application afin de donner un aperçu de comment les écrans s'affichent et s'utilisent.

4.4 Améliorations possibles

4.4.1 Ajout de fonctionnalités

Sweews possède des fonctionnalités qui en font un lecteur de flux d'actualité de base. Cependant, certaines fonctionnalités additionnelles permettraient de rendre l'application encore plus intéressante.

- **Lecture en mode hors-ligne**

  Il s'agit là d'une fonctionnalité qu'on retrouve fréquemment dans les applications d'actualité. En effet, les utilisateurs apprécient le fait de pouvoir télécharger au préalable toute l'actualité pour la consulter ensuite dans un lieu n'offrant pas d'accès à Internet. De base Sweews ne charge qu'à la volée le contenu des flux RSS. Comme nous l'avons vu, chaque fois que l'utilisateur lance l'application, les news sont chargées et à chaque fermeture elles sont effacées.

- **Importation vidéo**

  Durant le développement de l'application, Florian et moi-même avions pensé à l'importation de contenu audio-visuel, telles que des vidéos. La technologie Flash n'étant pas supportée par les appareils sous iOS, nous avions pensé à parser les pages Internet des journaux afin de localiser les vidéos via les balises « embed » pour les afficher ensuite dans une « webview » de l'application.

- **Recherche et ajout de flux RSS personnels.**

  Sweews étant un lecteur de flux d'actualité, il serait intéressant de pouvoir personnaliser les sources d'actualités en y ajoutant ses propres sources. Une fonction de recherche par exemple pourrait permettre de trouver d'autres flux.

- **Mode d'affichage par ligne**

  Au niveau ergonomique, il serait intéressant d'ajouter un autre mode d'affichage permettant de consulter les articles affichés horizontalement avec uniquement le titre et le texte. Cela donnerait la possibilité aux personnes n'étant pas à l'aise avec la « gridview » de se concentrer sur le contenu des articles.

- **Notifications push à l'utilisateur**

  Cette fonctionnalité serait une évolution des rubriques déjà implémentées. Elle permettrait de notifier l'utilisateur, lorsqu'il n'a pas l'application ouverte, des nouveaux articles traitant de ses rubriques.

4.4.2 Performances de l'application

Dans l'Objective-C, la gestion de la mémoire est un point essentiel à traiter tout au long du développement d'une application. En effet, chaque variable initialisée manuellement doit être gérée ensuite jusqu'à son effacement complet. Autrement, des fuites de mémoire ont lieu et les performances de l'application deviennent fragiles risquant un crash mémoire. C'est d'ailleurs un des points qui rend le langage difficile à appréhender pour un débutant.

J'ai tout au long du développement essayé d'appliquer les règles de gestion de mémoire, notamment en utilisant un outil qui analyse les performances et la mémoire utilisée par l'application. Voici un schéma de la mémoire allouée en cours d'utilisation de l'application :

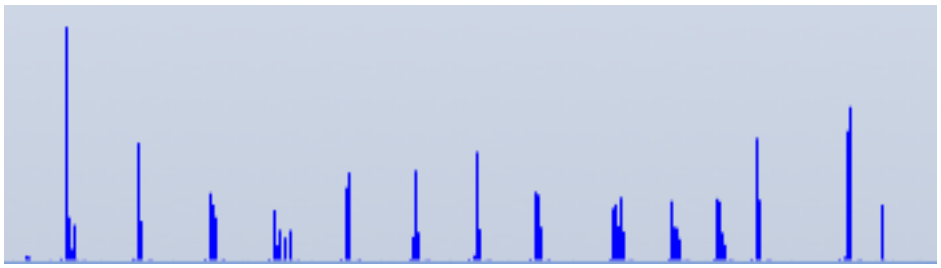


Figure 25 - Allocation de la mémoire

Les sommets bleus apparaissent lorsque nous changeons de catégorie et que les articles sont filtrés par catégorie. Ces pointes de mémoire sont dues principalement à l'initialisation des images et du contenu de chaque article à chaque changement de catégorie. Il serait intéressant de limiter un minimum toutes ces allocations afin de minimiser le total de mémoire utilisée par l'application.

Figure 26 - Utilisation de la mémoire

Ce deuxième graphique montre l'utilisation totale de la mémoire en cours d'utilisation de l'application. Les « crevasses » représentent la désallocation des objets lors du changement de catégorie. Cependant, nous remarquons que l'utilisation est sensiblement croissante malgré cette désallocation. Il semblerait que certains espaces mémoire ne sont pas libérés. Il serait donc judicieux d'analyser en détail les allocations afin d'optimiser l'utilisation de la mémoire et donc les performances globales de l'application.

5 DIFFICULTÉS RENCONTRÉES

Lors de la réalisation de ce travail de Bachelor, j'ai dû surmonter certains obstacles. Les principales difficultés découlaient des technologies utilisées et plus précisément de l'Objective-C, qui m'était encore inconnu en début de projet.

5.1 Le manque de connaissances

5.1.1 Compréhension de l'Objective-C

Une des principales difficultés dans le développement de l'application a été la compréhension de ce langage qui, comme expliqué à plusieurs reprises dans ce rapport, diffère des autres langages appréhendés tout au long de ma formation. En effet, l'Objective-C a une structure bien à lui, tant dans son fonctionnement que dans l'architecture même des objets qui le composent.

Il a donc fallu que je me forme en début de projet. Je dois avouer que sur la première semaine de formation, j'ai ressenti une certaine appréhension. Plus j'avançais dans mes lectures et tutoriels et plus je trouvais cette technologie déroutante. D'ailleurs, mes premiers tutoriels n'ont pas abouti, je n'arrivais pas à compiler les applications, il y avait toujours des erreurs qui apparaissaient et que je comprenais difficilement.

Au bout de la 2^{ième} semaine, il a malgré tout fallu que je commence le développement du prototype afin de ne pas prendre de retard dans mon planning. Avec recul, je pense que ce fut une erreur de ma part. J'aurais dû prendre plus de temps pour me former afin de pouvoir commencer dans les meilleures conditions.

Il est vrai, le développement du prototype et de l'application a pris plus de temps que prévu, notamment parce que mes connaissances étaient encore trop faibles à ce moment-là. Il m'arrivait en effet de recommencer l'implémentation d'une fonctionnalité qui marchait, certes, mais qui avait été développée de manière trop complexe ou erronée.

Pour conclure, je pense que cette difficulté a été un gros obstacle ralentissant ma productivité durant le développement. La planification effective en témoigne, avec les 50% du temps total de ce projet passé sur le développement. Si c'était à refaire, j'investirais plus de temps sur la formation en début de projet pour ainsi augmenter la productivité durant la phase de développement.

5.1.2 Gestion de la mémoire

En Objective-C, il faut constamment gérer les espaces mémoire alloués manuellement. Voici un petit exemple illustrant cette problématique :

```
-(void) setCouleur:(Couleur *) rouge
{
        couleur = rouge;
}
```

On désire ici attribuer une nouvelle couleur à l'objet « couleur »

Cet extrait de code marchera sans problème, cependant, l'objet initial sur lequel pointait la valeur « couleur » au début se retrouve sans pointeur, et ne sera donc plus libéré de la mémoire. Voici comment il faudrait procéder si l'on voulait correctement gérer la mémoire :

```
-(void) setCouleur:(Couleur *) rouge
{
    if(couleur != rouge)
    {
        [couleur release];
        [rouge retain];
        couleur = rouge;
    }
}
```

Ici on s'assure via le if que les deux pointeurs « rouge » et « couleur » ne pointent pas sur le même objet pour éviter de le supprimer avec le « release ». Ensuite, on libère l'objet sur lequel pointe « couleur » grâce à la méthode « release ». En faisant un « retain » de l'objet « rouge », on indique qu'un nouveau pointeur va l'utiliser. Enfin, on lui attribue ce pointeur.

Comme on le remarque, un simple changement d'attribut devient déjà plus complexe à développer. Il suffit maintenant d'imaginer l'application de cette logique derrière toute une fonctionnalité.

Or, c'est justement toute cette gestion des pointeurs et de la mémoire qui m'a fait ralentir durant la phase de développement. Ce qui, par exemple en java, se faisait en deux fois moins de lignes de code, devient tout de suite plus complexe.

5.1.3 Gestion de l'orientation de l'iPad

La problématique de l'orientation de l'iPad m'a posé passablement de problèmes. Il a en effet fallu prendre en compte cet élément pour chaque fonctionnalité et écran implémenté. Sur certaines vues, il a fallu afficher des éléments en fonction de l'orientation de l'iPad, car l'adaptation automatique ne fonctionnait pas. J'ai mis un moment à trouver une solution à ce problème :

```
if([[UIApplication sharedApplication] statusBarOrientation]
      == UIInterfaceOrientationPortrait)
{
    // mode portrait
else
{
    // mode paysage
}
```

Ce test permet de savoir dans quel mode d'orientation se trouve l'iPad et donc d'adapter ensuite les écrans.

D'ailleurs, l'adaptation était aussi peu évidente, car les éléments centrés au milieu de l'écran ne sont pas placés au même endroit que ce soit en mode paysage ou portrait. Il a donc fallu calculer le positionnement à chaque fois en fonction de l'orientation de l'iPad.

5.2 Le planning

En dehors du champ purement technique, une importante difficulté était due au planning de ce travail de Bachelor.

5.2.1 Jonglage avec les examens

En effet, pendant les 5 premières semaines de ce projet, il a fallu jongler entre les cours, les examens intermédiaires, les examens de modules et le travail de Bachelor.

De ce fait, les heures en début de projet n'étaient pas forcément respectées. Il était difficile par moment de se concentrer sur ce travail étant donné tout ce qu'il fallait réviser à côté.

6 CONCLUSION

6.1 Bilan général

6.1.1 Objectifs généraux

Par rapport aux objectifs généraux cités dans l'introduction de ce rapport, je pense que la plupart ont été remplis.

J'ai en effet effectué une analyse d'applications existantes. Mon seul regret est de n'avoir pas été plus en profondeur dans mon analyse afin d'en tirer des spécificités plus claires de notre propre application.

Par rapport à l'objectif de promotion de la HES-SO Valais, cette dernière aura, via cette application, un moyen de diffuser des informations et donc de se promouvoir. Et, concernant la promotion des acteurs de la presse nationale, Sweews donne la possibilité à l'utilisateur de choisir parmi une liste de journaux romands. Par ce biais, ces médias auront une chance de toucher un public cible plus large que leurs fidèles lecteurs. Une personne consultant d'habitude LeMatin, pourrait via Sweews aussi accéder aux informations du 20 Minutes en quelques secondes, sans devoir télécharger une tierce application.

Cependant, ces deux objectifs dépendront encore, dans un premier temps, de la validation de l'application par Apple pour sa mise en téléchargement sur l'AppStore et, dans un deuxième temps, du succès ou non de l'application.

Pour la modification des sources en tout temps, cet objectif n'était pas prioritaire en début de projet. Cependant, au fil du développement, avec Florian nous avons compris qu'il serait judicieux de pouvoir mettre à jour les sources de news sans devoir republier sur l'AppStore une mise à jour de l'application entière. La création d'un fichier source et son hébergement sur un serveur de l'HES-SO, m'ont permis de remplir cet objectif.

Donc, de manière générale, les principaux objectifs ont été remplis. Cependant, avec du recul, je pense qu'il aurait été important de garder un œil critique sur certains d'entre eux. En effet, si mon analyse des applications existantes avait été un peu plus poussée, nous aurions peut-être pu définir plus clairement les spécificités de l'application.

6.1.2 Objectifs technologiques

Par rapport aux défis technologiques, je pense qu'ils ont eux aussi été remplis de manière générale.

J'ai en effet développé une application iPad qui fonctionne sur l'iOS 4.3 d'Apple ainsi que sur les deux versions de l'iPad. Il faut par contre souligner que sur l'iPad 1ère génération, l'application est moins fluide. Je pense que ce problème est dû aux performances de l'appareil, plus faibles que la 2^{ième} version.

J'ai aussi cherché à rendre l'application ergonomique, notamment via une interface détaillée et la gestion de l'orientation de l'appareil.

Ensuite, malgré des soucis avec l'implémentation des méthodes asynchrones de l'application, je suis parvenu à intégrer un système de parsage des flux RSS qui permet de récupérer les articles des journaux.

6.2 Bilan personnel

6.2.1 Points positifs

De manière générale, j'ai trouvé ce projet très enrichissant. J'étais d'ailleurs très excité au moment de commencer, car je me réjouissais d'apprendre de nouvelles technologies et plus précisément le développement iOS. J'envisage, suite à ce cursus, me diriger vers ce domaine car ce type de développement m'intéresse grandement. Ce travail m'a permis de confirmer cette optique. Cependant, je reste conscient que j'ai encore passablement de lacunes à combler, notamment dans la gestion de la mémoire.

Je suis aussi plutôt satisfait de l'expérience acquise tout au long du projet. Que ce soit dans les aspects techniques (l'Objective-C, la base de données SQLite, l'administration d'un serveur IIS, l'utilisation de Mac OSX, etc.) ou la gestion de projet SCRUM, ce travail m'a permis d'étoffer mon bagage de connaissances.

6.2.2 Points à améliorer

Si je devais refaire le projet, avec recul je pense que je conduirais certaines étapes différemment.

D'abord, concernant la gestion de projet, j'essaierais de planifier de manière plus précise les tâches à effectuer. Je sais que via la méthodologie SCRUM il est difficile de planifier à l'heure exacte étant donné que le plan peut à tout moment changer et qu'il faut pouvoir s'adapter. Cependant, par moment, je restais trop vague dans mes estimations, ce qui m'a valu de faire des heures supplémentaires en fin de sprint (notamment sur les sprints 5 et 6). Après, il est vrai que je pouvais difficilement prévoir le temps passé sur le développement d'une fonctionnalité, car mes connaissances à ce stade ne me le permettaient pas. Aujourd'hui, je pense que je saurais un peu mieux jauger le temps à disposition.

Un autre point à améliorer serait de se concentrer sur les choses importantes et oublier les détails. Il est vrai que par moment, j'avais tendance à me focaliser sur un petit bug ou problème trop longtemps. Du coup, le reste de l'implémentation se voyait retardée. Exemple : lors de l'intégration de l'interface graphique, il m'est arrivé de passer plusieurs heures dans la création du bouton des rubriques qui ne fonctionnait pas entièrement. Au lieu de passer à la suite et d'y revenir plus tard, je me suis focalisé sur ce problème et ai perdu passablement de temps.

Enfin, le dernier point important à améliorer, serait de se former au préalable un peu plus sur la gestion de la mémoire. J'ai en effet essayé de traiter ce point tout au long du projet, cependant, je pense que je ne maîtrise pas encore bien tous les aspects de ce domaine qui sont pourtant essentiels dans le développement iOS.

7 SYNTHÈSE

Pour conclure ce rapport, je suis plutôt satisfait du travail accompli tout au long de ce projet. Malgré un manque de connaissances en début de projet, j'ai su me former tout en avançant dans le projet. Cependant, comme évoqué plus haut dans ce rapport, j'aurais dû me laisser une semaine de formation en plus afin d'avoir les bagages nécessaires pour la phase de développement. Cette phase m'a en effet pris plus de la moitié du temps consacré à ce travail de Bachelor. Je pense qu'avec les connaissances adéquates, le temps économisé aurait été assez conséquent. Malgré cette erreur de jugement, je suis satisfait de l'application dans son état actuel. Comme nous l'avons vu, elle remplit les principaux objectifs fixés.

Évidemment, elle gagnerait à être améliorée sur certains points. Le mode lecture ou la lecture de contenus vidéo seraient un plus non négligeable que l'utilisateur accueillerait certainement avec plaisir. Les performances de l'application et notamment l'utilisation de la mémoire de l'iPad, sont aussi deux éléments qu'il faudrait peut-être envisager d'optimiser dans une éventuelle future version.

Au niveau personnel, je pense que ce projet m'a permis d'acquérir des connaissances précieuses, qu'elles soient technologiques, techniques ou managériales. J'ai en effet fait l'expérience de la gestion SCRUM. Certes, la véritable expérience de cette méthodologie se fait dans un projet de groupe. Cependant, je pense que le fait de devoir planifier, gérer et maintenir un backlog et des sprints m'a permis de mieux cerner cette méthode. La planification initiale a certes été laborieuse, cependant je pense que j'ai su gérer efficacement les changements de direction du projet en adaptant les sprints en cours. Ce qui m'a permis de finir à temps l'application et remplir les objectifs principaux.

Enfin, ce travail m'a confirmé mon envie de continuer dans ce domaine de développement iOS. Malgré la difficulté d'appréhension de l'Objective-C, je trouve cet environnement OSX très attrayant. En effet, tout est très bien pensé et abouti, que ce soient les outils comme Xcode ou le simulateur iPad, le développement reste un plaisir. Cependant, je sais qu'il me reste certaines lacunes à combler, notamment dans la gestion de la mémoire, pour devenir plus efficace et créer des applications performantes.

8 TABLE DES ILLUSTRATIONS

9 REMERCIEMENTS

Je tiens à remercier mon professeur responsable de projet, Florian Doche, pour l'encadrement, sa disponibilité et ses conseils, tout au long de ce travail de Bachelor.

Je remercie aussi Guillaume Fort pour ses conseils sur la conception d'applications iOS et pour m'avoir inscrit à l'Apple Developer Program de la HES-SO; Jim Dovey pour sa disponibilité et son API AQGridView ; François Morard pour son aide dans l'administration du serveur, Gary Grand pour ses critiques et idées d'améliorations et Xavier Delpouve pour la relecture de ce rapport.

Pour finir, je remercie la HES-SO Valais et le service informatique pour avoir mis à disposition une salle, un poste de travail, et un iPad de test.

10 ATTESTATION

Je déclare, par ce document, que j'ai effectué le travail de diplôme ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de diplôme, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré.

Sierre, le 15 août 2011 Jonathan Moreira :

11 SOURCES

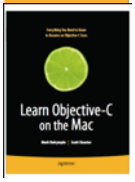
11.1 Livres et documents

Jeff LaMarche, Dave Mark. *Beginning iPhone 3 Development Exploring the iPhone SDK*, APress (2011)

Date de dernière consultation : 3 juin 2011

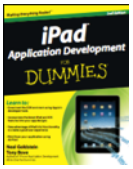
Jack Nutting, Jeff LaMarche, Dave Mark. *Beginning iPhone 4 Development: Exploring the iOS SDK,* APress (2011)

Date de dernière consultation : 5 juin 2011



Dalrymple Mark, Scott Knaster. *Learn Objective-C on the Mac,* APress (2009)

Date de dernière consultation : 23 juin 2011



Neal Goldstein, Tony Bove. *iPad Application Development for Dummies 2nd edition*, Wiley Publishing Inc. (2010)

Date de dernière consultation : 18 juin 2011

Dan Pilone, Tracey Pilone. *Head First iPhone & iPad Development*, O'REILLY. (2011)

Date de dernière consultation : 15 juillet 2011

11.2 Ressources web

- http://www.raywenderlich.com, par Ray Wenderlich, site de tutoriels sur le développement iOS. Date de dernière consultation : 25 juillet 2011
- http://mobile.tutsplus.com/tutorials/iphone/iphone-core-data/, tutoriel sur l'intégration de CoreData à un projet, date de dernière consultation : 12 juillet 2011
- http://www.blooberry.com/indexdot/html/topics/urlencoding.htm, par Brian Wilson, convertisseur d'URL, date de dernière consultation : 28 juin 2011
- http://stackoverflow.com/, consultations de divers sujets, date de dernière consultation : 6 août 2011
- http://www.siteduzero.com/tutoriel-3-176943-developper-sous-os-x-avec-cocoa.html?all=1, date de dernière consultation 14 juin 2011

11.3 APIs utilisées

- AQGridView de Jim Dovey, https://github.com/AlanQuatermain/AQGridView
- NSDate de Michael Waterfall, https://gist.github.com/953664
- ASIHTTPRequest de Ben Copsey, http://allseeing-i.com/ASIHTTPRequest/
- GDataXML de Google, http://code.google.com/p/gdata-objectivec-client/downloads/list
- GradientButtons de Jeff LaMarche, http://iphonedevelopment.blogspot.com/2010/05/programmatic-gradient-buttons.html
- NSArray+Extras.h de Ray Wenderlich, http://blog.jayway.com/2009/03/28/adding-sorted-inserts-to-uimutablearray/, adapté de Fredrik Olsson.